

METHOD FOR CONTROLLING STORAGE DEVICE

Patent Number: JP2001188712

Publication

date: 2001-07-10

Inventor(s): KOMATSU SHINPEI; MORITA YUMI; HAYASHI
TOMOHIRO; SHIBAZAKI SHOGO; ITO HIROYUKI;
TAKEHARA MASARU

Applicant(s): FUJITSU LTD

Requested

Patent:  JP2001188712

Application

Number: JP20000347526 19931001

Priority Number

(s):

IPC

Classification: G06F12/16; G11C16/02; G11C16/06; G11C29/00

EC

Classification:

Equivalents: JP3390740B2

**Abstract**

PROBLEM TO BE SOLVED: To improve the reliability of a memory in which data cannot be overwritten nor erased by byte units.

SOLUTION: When writing data are transmitted from a main boy processor 2 to a storage device 1, a control means 1b decodes an address by referring to a decoder, and writes data in the new sector or block of a storage area 1a. The decoder is constituted of two stags, and when part of the storage area 1a of the storage device 1 is destroyed, or when part of decoders 1d or 1e is destroyed, one or both of the decoders 1d and 1e in two stages is rewritten. Thus, decoding to the destroyed part is not performed, and even when a storage medium having possibility that part of it is destroyed, such as an EEPROM, a flash memory or the like, is used as the decoder, any probability that the address is turned into an error can be sharply reduced.

Data supplied from the esp@cenet database - 12

(11)特許出願公開番号

特開2001-188712

(P2001-188712A)

(43)公開日 平成13年7月10日(2001.7.10)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード [*] (参考)
G 0 6 F 12/16	3 1 0	G 0 6 F 12/16	3 1 0 Q
			3 1 0 P
G 1 1 C 16/02		G 1 1 C 29/00	6 0 1 B
16/06			6 0 1 C
29/00	6 0 1		6 3 1 D
		審査請求 有	請求項の数 1 〇 L (全 79 頁)
			最終頁に続く

(21)出願番号	特願2000-347526(P2000-347526)	(71)出願人	000005223
(62)分割の表示	特願平5-246547の分割		富士通株式会社
(22)出願日	平成5年10月1日(1993.10.1)		神奈川県川崎市中原区上小田中4丁目1番1号
		(72)発明者	小松 新平
			神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内
		(72)発明者	森田 由美
			神奈川県横浜市中区本町4丁目36番地 株式会社富士通コンピュータテクノロジ内
		(74)代理人	100100930
			弁理士 長澤 俊一郎 (外1名)

最終頁に続く

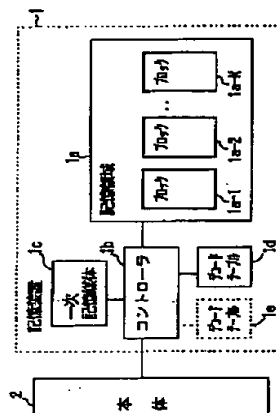
(54) 【発明の名称】 記憶装置の制御方法

(57) 【要約】

【課題】 上書きができず、また、バイト単位で消去することができないメモリの信頼性を向上させること。

【解決手段】 本体処理装置 2 から記憶装置 1 に書き込みデータが送られてくると、制御手段 1 b はデコーダを参照してアドレスをデコードし、データを記憶領域 1 a の新たなセクタもしくはブロックに書き込む。デコーダは 2 段設けられ、記憶装置 1 の記憶領域 1 a の一部が破壊したとき、もしくは、デコーダ 1 d、1 e の一部が破壊したとき、2 段のデコーダ 1 d、1 e のいずれか一方、もしくは、両方を書き換える。これにより、破壊した部分へのデコードが行われず、デコーダとして、EEPROM、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができる。

本発明の全体の概略構成を示す原図図



【特許請求の範囲】

【請求項1】 読み書きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域(1a)と、記憶領域(1a)中のデータが記憶されている場所を示す書き換え可能なデコーダ(1d)とを備えた記憶装置(1)の制御方法であって、

デコーダを2段設け、記憶装置(1)の記憶領域(1a)の一部が破壊したとき、もしくは、デコーダ(1d, 1e)の一部が破壊したとき、

2段のデコーダ(1d, 1e)のいずれか一方、もしくは、両方を書き換えることにより、破壊した部分へのデコードが行われなくようにしたことを特徴とする記憶装置の制御方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はパソコンの外部記憶装置等に使用されるフラッシュ・メモリ等の書き換えできない記憶装置の制御方法に関する。

【0002】

【従来の技術】近年、パソコン等の外部記憶装置として、フラッシュ・メモリを使った外部記憶装置が注目をあびている。フラッシュ・メモリは、不揮発性のためバックアップ電源が不要であり、電気的に書き換えることができ、しかも安価であるが、次のような問題点を持っている。

① データを消去してからでなければ、データの書き込みはできず、また、消去の単位はバイト単位は不可能であり、セクタ、ブロックもしくはチップ単位となる。このため、通常のメモリのように、バイト単位でデータを書き換えることはできず、読み出し速度に対して、書き込み速度あるいは消去速度が遅い。

② 消去回数に制限があり、通常、消去回数が10万～100万回程度で消去不能となるといわれている。このため、セクタ、もしくはブロックの消去回数が平均化するように使用しないと、消去回数の多い部分が先に不良となり、使用可能領域が減少する。フラッシュ・メモリは上記のような問題点をもっているため、フラッシュ・メモリを使用するにあたっては、データの退避領域を用意し、データの書き換え時にデータを退避したり、管理テーブルを設けてデータの書き込み／消去を管理したり、さらに、不良セクタ、ブロック等が発生した場合に救済措置を講ずるなど、種々の対策が講じる必要がある。

【0003】

【発明が解決しようとする課題】フラッシュ・メモリは不揮発性で電気的に書き換えることができる等の長所を持っている反面、上記のように多くの問題点を持っており、従来から使用されているDRAM、SRAM等の半導体メモリとは異なり、その使用にあたって解決しなければならない問題点も多い。本発明は上記した問題点に

鑑みなされたものであって、本発明の目的は、フラッシュ・メモリ等のように書き換えができず、また、バイト単位で消去することができないメモリの信頼性を向上させることである。

【0004】

【課題を解決するための手段】図1ないし図4は本発明の原理図であり、図1は本発明の全体の概略構成を示す原理図、図2～図4は本発明の概要を示す原理図である。図1ないし図4において、1は記憶装置、1aは書き換えすることができず、また、セクタ、ブロック等の所定の単位でしかデータを消去することができない、例えばフラッシュ・メモリ等から構成される記憶領域、1bは記憶領域1aへのデータの書き込み等を制御したり、記憶領域1aの消去等を行う制御手段、1cは記憶領域1aへのデータの書き込み時に一時的にデータを格納したり、また、記憶領域1aのデータを退避するための一次記憶媒体、1d、1eは記憶領域1aのアドレスをデコードするためのデコードテーブル、2は本体処理装置である。上記課題を解決するため、本発明の請求項1の発明は、図1、図4に示すように、読み書きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域1aと、記憶領域1a中のデータが記憶されている場所を示す書き換え可能なデコーダ1dとを備えた記憶装置において、デコーダを2段設け、記憶装置1の記憶領域1aの一部が破壊したとき、もしくは、デコーダ1d、1eの一部が破壊したとき、2段のデコーダ1d、1eのいずれか一方、もしくは、両方を書き換えることにより、破壊した部分へのデコードが行われなくようにしたものである。

【0005】

【作用】図1において、本体処理装置2から記憶装置1に書き込みデータが送られてくると、制御手段1bはデータを一次記憶媒体1cに格納する。そして、デコード・テーブル1d(1e)を参照してアドレスをデコードし、一次記憶媒体1cに格納されたデータを記憶領域1aの新たなセクタもしくはブロックに書き込む。その際、書き込むデータの論理アドレスと同一の論理アドレスのデータが記憶領域1aに既に存在している場合には、そのデータを消去したり、そのデータを格納した領域に消去可能フラグを立てる。また、データを格納した領域に消去フラグを立てた場合には、制御手段1bは所定の時期に所定の単位で、消去不可のデータを一次記憶媒体1cに退避して記憶領域1aを消去し、一次記憶媒体1cに退避したデータを記憶領域1aに書き戻す等の処理を行い、空き領域の作成等を行う。さらに、制御手段1bは記憶領域1aの一部が不良になった場合には、デコード・テーブル1d(1e)を書き換えるなどして、不良部分を予備領域で代替する。

【0006】図2(a)に示す記憶領域1aが複数のブロック1a-1, ..., 1a-Nと、該ブロックを区切っ

たセクタから構成され、書き込まれたデータをブロック単位で消去する記憶装置、図4に示す記憶領域1が上書きできない複数のセクタで構成され、セクタに書き込まれたデータをセクタ単位で消去する記憶装置において、データの書き込み/消去は次のように行われる。

(1) 記憶領域1aが複数のブロック1a-1, ..., 1a-Nと、該ブロックを区切ったセクタから構成され、書き込まれたデータをブロック単位で消去する記憶装置1において、図2(b)に示すように、ブロック1a-1, ..., 1a-N内の消去不可のデータを空きセクタがある他のブロックに退避するに際して、消去可能なセクタ数が多いブロックからmブロックと、消去可能な少ないブロックからnブロック(m, nは任意の整数)を選択し、選択されたブロックを同時に空きセクタがある他のブロックに、セクタを散在させて均一に、退避する。これにより、空きブロックを作成できるとともに、結果的に各ブロックに書き換えが多いセクタと書き換えが少ないセクタが混在することとなり、ブロックの消去回数を平均化することができる。

(2) 記憶領域1aが、退避領域を含む複数のブロック1a-1, ..., 1a-Nから構成され、書き込まれたデータをブロック単位で消去する記憶装置において、図2(c)に示すように消去可能なデータを含むブロック1a-1, ..., 1a-Nのデータの内、消去不可のデータのみを退避領域に移動したのち、上記ブロックのデータを消去し、消去したブロックを新たな退避領域とすることにより、空き領域を作成する。これにより、消去可能データをなくすことができ、本処理を予め行っておくことにより、書き込み時間を短縮することができる。

(3) データの書き込み方向に対して、退避領域の反対側のブロックの消去不可のデータのみを退避領域に移動したのち、上記ブロックのデータを消去し、消去したブロックを新たな退避領域とする。これにより、書き込み場所を示す書き込みポイントと退避領域の間に全ての空き領域を存在させることができる。したがって、処理の中断があっても、書き込みポイントの示す位置からデータを書き込むことができ、処理中断後に必要とする処理量を少なくすることができる。

(4) 記憶領域1が、予備領域を含む複数のブロック1a-1, ..., 1a-Nから構成され、書き込まれたデータをブロック単位で消去する記憶装置において、図2(a)、図3(a)に示すように書き込みポイントを予備領域に設定し、データが書き込まれたブロックの消去不可のデータを予備領域に移動したのち、上記ブロックのデータを消去する処理を繰り返し、最後に消去したブロック以外のブロックの空きセクタ数が予備領域にあるセクタ数と等しいか、もしくは、大きくなったとき、予備領域のデータを最後に消去したブロック以外のブロックに移動する。これにより、空き領域を作成することができる。また、空きブロックが使用不可になったとき、

使用不可となったブロックを救済して空きブロックを作成することができる。さらに、予備領域のデータを、データを最後に消去したブロック以外のブロックに移動する際、移動先の書き込みポイントを、データを最後に消去したブロックの後のブロックから検索して予備領域のデータを移動するようにすることにより、最後に消去をかけたブロックと書き込みポイントの間に全ての空きセクタを存在させることができ、処理中断後の処理を簡単にすることができる。またさらに、処理中断後、処理を再開したとき、予備領域に消去可能なデータがあるか、あるいは、消去不可のデータのみがあるかに応じて、処理中断時点と判断することができ、処理中断時点の判断を単純化することができる。

(5) 記憶領域1が複数のブロック1a-1, ..., 1a-Nから構成され、書き込まれたデータをブロック単位で消去する記憶装置1において、ブロック毎に書き込みの有無を示すフラグF1を設けて、図3(b)に示すように、フラグF1を参照して、書き込まれたデータを消去する。これにより、不要な消去を避けることができ、消去時間の短縮を図ることができるとともに、記憶媒体の寿命を延ばすことができる。また、全記憶領域の消去処理が実行中であることを示す実行中フラグF2を設けることにより、全記憶領域の消去処理を実行中に、処理が中断された際、実行中フラグを参照して、全記憶領域の消去処理中であることを判断することができ、不意の処理中断に対応することができる。

(6) 記憶領域1a上に同一の機能に対応した複数ビットのフラグを記録し、該フラグの論理積によりフラグの判定を行うようにする。これにより、全ビットが不良にならない限り、正しい判定値を常に出力することができ、システムの信頼性を向上させることができる。

(7) 図4に示すように、記憶領域1が上書きできない複数のセクタで構成され、セクタに書き込まれたデータをセクタ単位で消去する記憶装置において、同一のセクタ番号を持つセクタを少なくとも2個用意し、同一番号が付されたセクタのいずれか一方にデータがあった場合、他方のセクタにデータを書き込むと同時に、一方のセクタのデータを消去する。これにより、書き込み速度を向上させることができる。

(8) 図4に示す記憶装置において、書き込みするセクタにデータがある場合、データを一次記憶装置1cに転送している間に、上記セクタのデータを消去し、データの消去完了後、もしくは、データ転送の終了後、上記セクタにデータを書き込む。これにより書き込み速度を向上させることができる。

(9) 図4に示す記憶装置において、不良なセクタを代替する予備の領域1b-1, 1b-2と、書き込みセクタを管理する書き換え可能なデコード・テーブル1dを設け、不良なセクタが発生したとき、上記デコード・テーブル1dを書き換えることにより、予備の領域1b-

1, 1b-2に書き込みセクタを変更し、予備の領域1b-1, 1b-2がなくなつたとき、デコード・テーブル1dを書き換えてセクタを再構成するようにしたので、不良セクタ発生時の救済を行うことができる。

(10)図4に示す記憶装置において、N個のセクタ番号に対してN+1個のセクタを設けるとともに、N+1個の各セクタをN個のセクタ番号で共有し、データをセクタへ書き込む際、データをN+1個のセクタの内の空いているセクタに書き込む。これにより、書き込み速度を向上することができるとともに、余分に用意するセクタの数を少なくすることができ、コストダウンを図ることができる。また、不良なセクタを代替する予備の領域1b-1, 1b-2を設け、不良なセクタが発生したとき、デコード・テーブル1dを書き換え、予備の領域1b-1, 1b-2に書き込みブロックあるいはセクタを変更するようにしたり、デコード・テーブル1dを再編成する。これにより、不良セクタ発生時の救済を行うことができる。

(11)記憶領域1aが複数に分割されており、分割されたチップ内に複数のブロックが設けられ、チップ内のブロックにデータを書き込む際、新たなブロックにデータを書き込み、書き込んだデータがチップ内に既に存在している場合には、そのデータをブロック単位で消去する記憶装置において、データを書き込むチップを、書き込むデータに対応させて固定する。これにより、制御手段1bは各チップ内のデータのみを管理すればよく、全領域を管理する必要がない。このため、制御手段1bにおける管理テーブルを簡単にすることができ、書き込み速度を向上させることができる。また、各チップ内にデータを書き込むための複数のワークブロックを設けることにより、ワークブロックに不良が発生した場合に、不良となったワークブロックを救済することができる。

【0007】本発明の請求項1の発明は、上述の読み書きが可能で、かつ、記憶領域の一部が破壊する可能性のある記憶領域1aと、記憶領域1a中のデータが記憶されている場所を示す書き換え可能なデコーダ1dとを備えた記憶装置において、図4に示すようにデコーダを2段設け、記憶装置1の記憶領域1aの一部が破壊したとき、もしくは、デコーダ1d, 1eの一部が破壊したとき、2段のデコーダ1d, 1eのいずれか一方、もしくは、両方を書き換えることにより、破壊した部分へのデコードが行われないようにしたので、デコーダとして、EEPROM、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができ、信頼性を確保することができる。

【0008】

【実施例】次に、本発明の実施例を説明する。

A. 本発明の前提となる記憶装置の構成。

図5は本発明の前提となるフラッシュ・メモリを使用し

た記憶装置の構成を示すブロック図である。同図において、20は例えばメモリ・カード等の記憶装置、21はフラッシュ・メモリのデータの書換え、消去等を制御するコントローラLSI、22はプロセッサ、23は各種テーブルを格納したり、データの書き込み時のデータ・バッファあるいは退避バッファとして使用されるSRAM、24はクロック発振器、25-1~25-5はフラッシュ・メモリである。同図において、本体よりSRAM23に転送されたデータをフラッシュ・メモリ25-1~25-5に書き込む際、前記したようにフラッシュ・メモリは上書きができないので、データを書き込み場合には、新たな領域にデータを書き込む。また、データ更新の場合には、旧データに消去フラグを立てる等の措置を講ずる。そして、旧データに消去フラグが立てられている場合には、所定の時期に、消去フラグが付されていないデータのみを所定の単位でSRAM23の退避領域に退避し、フラッシュ・メモリの書き込むべき領域を消去したのち、データを書き戻すことによりデータの整理を行う。

【0009】図6は上記したSRAM23の内容を示す図であり、同図に示すように、SRAM23には下記のデータが格納される。

(1) 消去回数テーブル

フラッシュ・メモリの各ブロックの消去回数を保持している。

(2) 消去可能セクタ数テーブル

各ブロックの消去可能セクタ・フラグの立っている総数値を保持する。

(3) 書き込みポインタ

フラッシュ・メモリに書き込みを開始するチップNo., ブロックNo., セクタ・アドレスNo. を保持する。

(4) WORK-BLOCK-No.

現在のWORK-BLOCK-No. を示し、チップNo., ブロックNo. を保持する。

(5) 整理ポインタ

現在、整理をしているチップNo., ブロックNo., セクタ・アドレスNo. を保持する。

(6) 書き換え有無

現在、書き込みするデータが新規であるかどうかを示す。

(7) 書き込み回数

整理時の本体書き込み回数を管理する。

(8) 退避カウンタ

整理時の退避動作のセクタ数を管理する。

(9) チップ数

カードの搭載フラッシュチップ数を保持する。

(10)セクタマップ・テーブル

論理アドレスの変換用にチップNo., ブロックNo., セクタ・アドレスNo. を保持する。

【0010】図7、図8はフラッシュ・メモリ25-1

～25-5の内容を示す図であり、フラッシュ・メモリの各セクタには、次の(1)から(5)に示す制御情報、データ等が格納されている。なお、この例においては、セクタが126設けられており、各セクタには、(1)から(5)の管理情報等とデータが格納され、126番目のセクタの後に、次の(6)から(14)の管理情報等が格納されている。

(1) 不良フラグ

このセクタの状態を示し、使用不可の場合に、ここに書き込みを行う。

(2) 消去フラグ

このセクタのデータの状態を示し、書き換え等で無効になった場合に、ここに書き込みを行う。

(3) 論理アドレス

このセクタの論理アドレスを示す。

(4) データ

このセクタに書き込まれたデータ。

(5) チェックサム

書き込んだデータのチェックサム

(6) 不良セクタ・メモリ

整理対象ブロックの中の不良セクタを示す。

(7) 整理元消去回数

整理対象ブロックの消去回数。

(8) 退避ブロックNo.

データを退避してくるチップNo. , ブロックNo.

(9) 消去Start

整理対象ブロックの消去を開始する時に立てる。

(10) 消去End

整理対象ブロックの消去を終了するときに立てる。

(11) All Erase対象

All Erase時に消去対象として確定した時に立てる。

(12) 空きブロック

ブロック内にデータがあるかを示し、データがある場合に立てられる。

(13) ブロック・ステータス

このブロックの状態を示し、使用不可になった場合に立てる。

【0011】図9ないし図13はフラッシュ・メモリ25-1から25-5へのデータ書き込み時のプロセッサ22における処理を示すフローチャートであり、同図を参照して書き込み時の動作を詳細に説明する。図9のステップS1において、SRAM23の、現在書き込みをするデータが新規であるか否かを示す「書き換えの有無」(図6参照)を0として、ステップS2において、整理中であるか否かを判別する。整理中でない場合には、ステップS5に行き、SRAM23の、整理時の本体書き込み回数を管理する「書き込み回数」を0にしステップS7に行く。また、整理中の場合には、ステップS3に行き、「書き込み回数」に1を加え、ステップS

4において、書き込み回数が6であるか否かを判別する。書き込み回数が6の場合には、ステップS6に行く、書き込み回数を1にする。また、書き込み回数が6でない場合には、ステップS7に行く。上記処理は、後述するように、書き込み回数が2～5の場合には整理動作を行わないようし、書き込み回数が1の時のみ整理動作を行うことにより、頻繁な整理動作を避けるためであり、上記のように書き込み回数を設定することにより、5回に1回整理動作を行うこととなる。ステップS7において、書き込むデータの論理アドレスがオーバーしているか否かを判別し、オーバーしている場合には、エラー処理を行う。論理アドレスがオーバーしていない場合には、ステップS8に行き、旧データがあるか否か、すなわち、現在書き込むデータが新規なデータであるか否かを判別し、新規なデータでない場合には、ステップS9において、SRAM23の「書き換えの有無」を1にしてステップS11に行く。また、新規なデータの場合には、ステップS10において、SRAM23の「書き換えの有無」を0にしてステップS11に行く。ついで、ステップS11において、本体からデータをSRAMに転送し、ステップS12において転送エラーがあるか否かを判別する。転送エラーがある場合にはエラー処理を行う。転送エラーがない場合には、図10のステップS13に行き、書き換えの有無、すなわち、新規データであるかどうか判別し、新規データでない場合には、データの更新であるので、ステップS14において旧データのあるセクタに消去ビットを書き込み、ステップS15に行く。

【0012】ステップS15において、SRAM23の整理時の書き込み回数を管理する「書き込み回数」が1であるか否かを判別し、「書き込み回数」が1の場合には、ステップS16以降でフラッシュ・メモリの整理を行う。また、「書き込み回数」が1でない場合には、図13のステップS46に行き、ステップS46以降で、SRAM23のデータをフラッシュ・メモリへ書き込む。ステップS16において、SRAM23上の整理時における退避動作のセクタ数を管理する「退避カウンタ」を0にして、ステップS17において、現在整理をしているセクタアドレスNo.を示す「整理ポインタ」(セクタアドレス)が126セクタより多いか否かを判別し、126より小さくない場合には、整理が終了したものとして、図12のステップS38に行く。また、「整理ポインタ」(セクタアドレス)が126セクタより少ない場合には、ステップS18に行き、整理ポインタが0であるか否かを判別し、整理ポインタが0でない場合にはステップS22に行く。

【0013】整理ポインタが0の場合には、ステップS19において、各ブロックの消去回数や消去可能セクタ数をSRAM23のテーブルから求め、それをもとにして、整理対象を選択する。ついで、ステップS20にお

いて、フラッシュ・メモリ上のWORK-BLOCKの退避ブロックNo. (図8参照)に整理対象のチップNo.、ブロックNo.を書き込む。ステップS21において、書き込みエラーがあるか否かを判別し、書き込みエラーがある場合には、エラー処理を行い、書き込みエラーがない場合には、ステップS22に行く。ステップS22において、退避するデータを検索する。すなわち、フラッシュ・メモリのデータに消去フラグ(図7参照)が書き込まれている場合には、次のセクタに行く。また、論理アドレスが無い場合には消去フラグを書き込み、次のセクタに行き、論理アドレスが異常のものは消去フラグと不良フラグを書き込み、次のセクタに行く。

【0014】次いで、図11のステップS23に行き、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS24において、整理ポイント(セクタアドレス)が126番目のセクタを指しているか否かを判別し、整理ポイントが126である場合には、整理が終わったものとして、図12のステップS38に行く。整理ポイントが126でない場合には、ステップS25に行き、退避するデータをフラッシュ・メモリからSRAM23に移動し、ステップS26において、生成したチェックサムがフラッシュ・メモリ上のチェックサム(図7参照)と一致するか否かを判別する。一致しない場合には、ステップS27において、チェックサムがFFhであるか否かを判別する。そして、チェックサムがFFhでない場合には、ステップS28に行き、フラッシュ・メモリ上のチェックサムをFFhとし、データ退避元のセクタの不良フラグを書き込み、ステップS29に行く。ステップS29において、書き込みエラーが有るか否かを判別し、ステップS30に行く。また、ステップS26において、チェックサムが一致するか、S27において、チェックサムがFFhの場合には、ステップS30に行き、書き込みポイントが示すセクタから書き込めるセクタを探し、ステップS31において、書き込み可能セクタがあるか否かを判別する。書き込み可能セクタがない場合には、エラー処理を行い、書き込み可能セクタがある場合には、ステップS32に行き、SRAM23上のデータをフラッシュ・メモリに移動する。

【0015】ステップS33において、書き込みエラーが有るか否かを判別し、書き込みエラーがある場合には、ステップS30に戻る。書き込みエラーがない場合には、図12のステップS34に行き、データ退避元のセクタの消去フラグを書き込む。そして、ステップS35において、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS36において、SRAM23上のセクタマップテーブルを書き換えるとともに、退避カウンタに1を加える。ついで、ステップS38において、退避動作が予め定められた所定回数(この場合には60回)行われたか否かを判別し、退

避動作が60回になった場合には、ひとまずフラッシュメモリの整理を終えてステップS38に行く。また、退避動作が60回になっていない場合には、図10のステップS22に戻り上記処理を繰り返す。

【0016】退避カウンタが60になっているか、前記した図10のステップS17において、整理ポイントが126より小さくないと判別された場合には、ステップS38に行き、退避カウンタが0であるか否かを判別する。そして、退避カウンタが0である場合、つまり、整理ポイントが126より小さくない場合であって、かつ、退避動作が行われていない場合には、ステップS39に行き、ステップS39以降で、不良フラグの情報を不良セクタメモリに書き込む等の処理を行う。すなわち、退避カウンタが0である場合には、処理時間が短かったので、S39以降の不良フラグの情報を不良セクタメモリに書き込む等の処理を行い、また、退避カウンタが0でない場合には、図13のステップS46に行き、SRAM23上のデータをフラッシュ・メモリに移動する等の処理を行う。

【0017】ステップS39において、退避したブロックの不良フラグの情報を書き込み先(書き込み先はWORK-BLOCK)の不良セクタ・メモリ(図8参照)に書き込み、また、消去回数を整理元消去回数欄(図8参照)に書き込む。ステップS40において、書き込みエラーがあるか否かを判別し、書き込みエラーがない場合には、ステップS41に行き、整理したブロックを消去する。ステップS42において、消去エラーがあるか否かを判別し、消去エラーがない場合には、図13のステップS43に行き、消去したブロックに不良セクタメモリから不良フラグ情報を戻し、整理元消去回数に1を加えて、消去回数に書き込む。ついで、ステップS44において、消去したブロックに対応するテーブルの消去回数に1を加え、消去可能セクタ数を0にする。ステップS45において、整理ポイントをWORK-BLOCKに入れ、整理ポイントを0とする。

【0018】次にステップS46に行き、書き込みポイントの示すセクタから書き込めるセクタを探し、ステップS47において書き込み可能セクタが有るか否かを判別する。書き込み可能セクタがある場合には、ステップS48に行き、SRAM23上のデータをフラッシュ・メモリに移動し、ステップS49において、書き込みエラーがあるか否かを判別する。書き込みエラーが有る場合には、ステップS46に戻り、書き込みエラーがない場合には、ステップS50に行く。ステップS50において、書き換え有無が1であるか否か、すなわち、書き込みするデータが新規であるか否かを判別し、書き換えの有無が1の場合には、ステップS51で消去フラグを書き込んだセクタに対応する消去可能セクタ数テーブル(図6参照)の値に1を加え、ステップS52に行く。ステップS50において、書き換え有無が0である場合

には、ステップ52において、セクタマップテーブル（図6参照）を書き換え、ステップ53において、次の書き込みに備えて、書き込みポイントの示すセクタから書き込めるセクタを探しておき終了する。

【0019】B. 書き込み／消去処理。

前記したように、フラッシュ・メモリは、消去回数に制限があり、全てのメモリを有効に活用するためには、消去回数を平均化する必要がある。また、フラッシュ・メモリは上書きをすることができず、データを書き換える場合には、新たなセクタにデータを書き込んで、旧データに消去可能フラグを立ておくなどの措置を講じ、適宜の時点で消去可能フラグが立っているセクタを消去する必要がある。このため、消去可能なデータを消去してデータを書き込むための空き領域を確保する必要があるとともに、上記空き領域が、不良ブロックが発生するなどの原因により使えなくなったとき、救済措置を講じて新たな空き領域を作成する必要がある。以下に、上記した消去回数の平均化（実施例1）、消去可能なデータを消去して空き領域を確保する空き領域の作成（実施例2）、不良ブロック発生時の空き領域作成のための救済措置（実施例3）等の領域処理についての本発明の実施例を示す。

【0020】（1）実施例1（消去回数の平均化）

前記したように、フラッシュ・メモリは、消去回数に制限があり、全てのメモリを有効に活用するためには、消去回数を平均化する必要がある。本実施例は、各ブロックの消去回数を意識することなく、消去回数のバラツキを抑えることが可能な実施例を示しており、本実施例においては、セクタ単位で追記書き込み、ブロック単位で消去するメモリにおいて、ブロックをセクタ単位で区切り、以下に示すように、消去可能なセクタが多いブロックと、消去可能なセクタが少ないブロックとを混ぜ合わせて新しいブロックに書き込むことにより、消去回数を平均化している。次に本実施例について説明する。図14ないし図18は本実施例における書き込み処理を示す図であり、本実施例においては、6セクタを持つ6ブロックのフラッシュ・メモリにAからOの論理セクタを書き込む例を示している。また、以下の説明においては、ブロック内の空きセクタの数がN個（本実施例では2個）になった場合にM個のブロック（本実施例では2個）に退避動作を行うこととし、また、消去可能セクタ数が多い方からmブロック（本実施例では1ブロック）と消去可能セクタの少ない方からnブロック（本実施例では1ブロック）を選択して、同時に退避する。なお、退避動作を行う場合には、前記したように、消去可能フラグが立っていないセクタを読み出して、図5に示すSRAM23に移動し、ついで、SRAM23に移動したセクタを新たなブロックに書き込む。

【0021】図14の（a）において、同一のセクタがないので、ブロック1とブロック2にセクタA、B、

C、D、E、F、G、H、Iを、最初のセクタから順番に書き込む。（b）において、セクタC、Dを書き込む。この場合には、同一のセクタC、Dがあるので、既にC、Dが書き込まれているセクタに消去可能フラグを立て、新しいセクタC、Dを書き込む。ついで、図15の（c）において、セクタJ、K、L、M、N、Oを書き込む。この場合にも、（a）の場合と同様、同一のセクタがないので、既に書き込まれたセクタの後に順番に書き込む。（d）において、セクタH、I、J、K、M、Nを書き込む。同一セクタが書き込まれているので、消去可能フラグを立て、新しいセクタH、I、J、K、M、Nを書き込む。

【0022】図16（e-1）、（e-2）において、セクタC、Dを書き込む。その際、空きブロックの数が2個になっているので、退避動作を行う。すなわち、図15の（d）において、消去可能なセクタが最も多いブロック3と、消去可能セクタが最も少ないブロック4を選択し、ブロック3のセクタとブロック4のセクタを混在させ、退避先の各ブロックのセクタ数が均一になるように、空きブロック5と6に書き込む。その結果、図16（e-1）に示すように、セクタO、I、K、Mがブロック5に、また、セクタH、J、L、Nがブロック6に書き込まれ、ブロック3と4が空きブロックとなる。この状態で、既にC、Dが書き込まれたブロック2に消去可能フラグを立て、セクタC、Dを図16（e-2）に示すようにブロック5に書き込む。

【0023】図17（f-1）において、セクタH、I、Jを書き込む。この場合にも、同一のセクタがあるので、ブロック5とブロック6に消去可能フラグを立て、ブロック6にセクタH、Iを書き込むが、セクタJを書き込む時点で空きブロックが2個になったので、前記と同様退避動作を行う。すなわち、（f-2）に示すように、消去可能なセクタが最も多いブロック2と、消去可能セクタが最も少ないブロック6を選択し、ブロック2のセクタとブロック6のセクタを混在させ、退避先の各ブロックのセクタ数が均一になるように、空きなブロック3と4に書き込む。ついで、（f-3）に示すように、セクタJを書き込む。この場合にも同一セクタがあるので、既に書き込まれているブロック4のセクタに消去可能フラグを立て、ブロック3にセクタJを書き込む。

【0024】図18（g）において、セクタE、F、Gを書き込む。この場合にも同一セクタがあるので、既に書き込まれているブロック1、3のセクタに消去可能フラグを立て、ブロック3と4にセクタE、F、Gを書き込む。図19は本実施例の処理を示すフローチャートであり、同図により本実施例の処理について説明する。ステップS1において、本体よりデータを受け取ると、ステップS2において、空きセクタ有りのブロック数がN以下であるか否かを判別する。空きセクタ有りのブロック

数がN以下でない場合には、ステップS7に行く。空きセクタ有りのブロック数がN以下の場合には、ステップS3において、消去可能セクタ数が多いブロックからmブロックを退避対象とし、ステップS4において、消去可能セクタ数が少ないブロックからnブロックを退避対象とする。ステップS5において、退避対象のブロックからデータを空きブロックに移動する。その際の書き込み方は、データを移動する毎に書き込みブロックを変え、M個のブロックに書き込んだら最初に書き込んだブロックに戻る。ついで、ステップS6において、退避対象のブロックを消去する。ステップS7において、同一論理セクタがあれば、既に書き込まれている物理セクタに消去可能フラグを立て、ステップS8において、本体から受け取ったデータをフラッシュ・メモリに書き込む。

【0025】ところで、一般にセクタには、システム・プログラムのように殆ど書き換えが行われないデータを格納したセクタと、常時書き換えが行われるデータを格納したセクタがあり、普通にデータの書き込み/退避をおこなっていると、殆ど書き換えが行われないセクタが多いブロックは消去回数が少なくなる。したがって、上記のように、消去可能なセクタが多いブロック（常時書き換えが行われるセクタが多いブロック）と、消去可能なセクタが少ないブロック（殆ど書き換えが行われないセクタが多いブロック）とを混ぜ合わせて新しいブロックに書き込むことにより、結果的に各ブロックに書き換えが多いセクタと書き換えが少ないセクタが混在することとなり、ブロックの消去回数を平均化することができる。本実施例においては、上記のような原理に基づき書き込み処理をおこなっているため、各ブロックの消去回数を意識することなく、消去回数のバラツキを抑えることができ、フラッシュ・メモリのように消去回数が有限なメモリの寿命を延ばすことが可能となる。また、消去回数をカウントすることなく消去回数のバラツキを抑えることができるので、消去回数をカウントするための領域を設けることなく、メモリの管理を行うことも可能となる。

【0026】(2) 実施例2（消去可能データの消去による空き領域の作成）

前記したように、フラッシュ・メモリは上書きをすることができず、消去した後でなければ書き込みを行うことができない。このため、書き込み時、同一のセクタがある場合には消去可能フラグを立て、適宜の時点で消去可能フラグが立っているセクタを消去する必要がある。本実施例は、上記のように、消去フラグがたっている消去可能なデータを記憶媒体上からなくすための領域処理を示しており、本実施例による領域処理を処理の空き時間などに予め行っておくことにより、書き込みできる領域を確保でき、書き込み時間を短縮することができる。

【0027】図20から図23は本実施例における書き

込み/消去処理を示す図であり、同図により本実施例を説明する。なお、本実施例においては、次の点を前提としている。

- ① ブロック数が6個で、1ブロックあたり、6個のセクタがあり、リード/ライトはセクタ単位で行い、消去はブロック単位で行う。
- ② 書き込み方式は追記書き込み方式であり、同じアドレスの論理セクタを書き込むときは、旧論理セクタに格納されている物理セクタに消去可能フラグを立てる。
- ③ 追記のために書き込み領域を5ブロック+退避領域1ブロック（ブロック1～ブロック6）とする。
- ④ 書き込み領域がなくなった場合には、消去可能フラグが立っている物理セクタが一番多いブロックの消去不可の物理セクタのデータを退避領域に移動して、移動元のブロックを消去する。そして、そのブロックを退避領域として使用し、今までの退避領域を書き込み領域とする（この一連の動作を退避動作と呼ぶ）。
- ⑤ 本体から送られて来るセクタのアドレスはAからPとする。
- ⑥ ブロック中に消去可能なデータがm以上あるとき、本実施例における領域処理の対象とする（図20から図23の例においては、 $m=1$ としている）。なお、図20から図23中のブロックの右に付された矢印は書き込み場所を示す書き込みポイントの位置である。

【0028】図20(a)において、論理セクタA～Gを書き込む。この場合には、同一アドレスの論理セクタがないので、書き込みポイントが指す位置から順番に書き込む。(b)において、論理セクタA、D～Gを書き込む。この場合には、同アドレスの論理セクタ(A、D、E、F、G)があるので、消去可能フラグを立てた後に、(a)と同様に書き込む。(c)において、論理セクタH～Nを書き込む。この場合には、同一アドレスの論理セクタがないので、そのまま書き込む。図21(d)において、A、G～Lを書き込む。この場合には、同アドレスの論理セクタ(A、G、H、I、J、K、L)があるので、消去可能フラグを立てた後に書き込む。(e)において、A、H～Jを書き込む。この場合にも、同アドレスの論理セクタ(A、H、I、J、)があるので、消去可能フラグを立てた後に書き込む。

(f-1)において、A、I～Lの書き込みを行おうとするが書き込み領域がないので、退避動作を行う。すなわち、消去可能フラグが立っている物理セクタが一番多いブロック（ブロック3）のデータを退避領域（ブロック6）に移動して、移動元のブロックを消去する。そして、そのブロック（ブロック3）を退避領域として使用し、今までの退避領域（ブロック6）を書き込み領域とする。その結果、(f-1)のようになる。

【0029】ついで、図22(f-2)に示すように、A、I～Lを書き込む。この場合には、同アドレスの論理セクタがあるので、消去可能フラグを立てた後にA、

I～Lを書き込む。この時、書き込みポインタは退避領域（ブロック6）の先頭にある。以上のような書き込みにより、消去可能なデータを増えたので、本実施例による領域処理を行う。すなわち、退避領域（ブロック3）から見て、書き込み方向に対して一つ反対方向のブロック（ブロック2）を処理対象として、ブロック2と3で退避動作を行う。その結果、図22（g-1）に示すように、ブロック2が退避領域となり、ブロック3にブロック2の消去不可のデータが移される。次に上記と同様に、ブロック1とブロック2で退避動作を行う。その結果（g-2）に示すように、ブロック1が退避領域となり、ブロック2にブロック1の消去不可のデータが移される。

【0030】ついで、上記と同様にブロック5と1で退避動作を行う。その結果、図23（g-3）に示すように、ブロック5が退避領域となり、ブロック1にブロック5の消去不可のデータが移される。なお、ブロック6には消去可能なデータがないので、領域処理の対象としない。同様にブロック4と5で退避動作を行う。その結果、（g-4）に示すように、ブロック4が退避領域となり、ブロック5にブロック4の消去不可のデータが移される。そして、退避領域の位置が処理を行う前の退避領域の位置まできたので、領域処理を終わる。以上のような処理を行うことにより、消去可能データは記憶媒体上からなくなる。次に（h）に示すように、A、H～Jを書き込む。この場合にも、同アドレスの論理セクタがあるので、消去可能フラグを立てたのち書き込む。

【0031】図24は本実施例の処理を示すフローチャートであり、同図により本実施例の処理を説明する。ステップS1において、現在の退避領域のブロックNo.を記憶し、ステップS2において、退避領域を処理対象とする。ステップS3において、処理対象を書き込み方向とは反対方向に一つ進める。ステップS4において、処理対象とステップS1において保存した保存ブロックNo.を比較し、同じであれば処理を終了する。処理対象と保存ブロックNo.が異なる場合には、ステップS5に行き、処理対象の消去可能セクタ数を所定値mと比較し、消去可能セクタ数がm以下の場合には、ステップS3に戻り、上記処理を繰り返す。消去可能セクタ数がmより大きい場合には、ステップS6に行き、処理対象を退避対象として、ステップS7において、書き込みポインタを退避領域の先頭に移動する。ステップS8において、退避対象のブロックからデータを移動する。ステップS9において、退避対象のブロックを消去し、ステップS10において、退避対象のブロックを退避領域とし、ステップS3に戻り上記処理を繰り返す。

【0032】本実施例においては、上記のようにして領域処理を行っているので消去可能データを無くすことができ、本実施例の領域処理を予めおこなっておくことにより、書き込み時間の短縮を図ることができる。また、

領域処理の対象となるブロックを書き込み方向とは反対方向に進めているので、書き込み場所を示している書き込みポインタと退避領域の間に全ての空き領域を存在させることができる。その結果、どのような場合に処理の中断がおこっても、書き込みポインタの示す位置からデータを書き込んで行くことができ、処理中断後に必要とする処理を少なくすることができ、また、処理を全ておえていなくても、支障がない。

【0033】（3）実施例3（空き領域の作成のための救済措置）

前記したように、フラッシュ・メモリはデータ書き込みのため、メモリ中に空き領域を確保しておく必要があるが、本実施例は、上記空き領域が、不良ブロックが発生するなどの原因により使えなくなったとき、救済措置を講じて新たな空き領域を作成する実施例を示している。図25から図31は本実施例の空き領域の作成処理を示す図であり、同図により本実施例を説明する。なお、本実施例においては次の点を前提としている。

- ① ブロック数が7個で、1ブロックあたり、6個のセクタがあり、リード／ライトはセクタ単位で行い、消去はブロック単位で行う。
- ② 書き込み方式は追記書き込み方式であり、同じアドレスの論理セクタを書き込むときは、旧論理セクタに格納されている物理セクタに消去可能フラグを立てる。
- ③ 障害時の救済用に1ブロックの予備領域（ブロック0）を確保し、追記のために書き込み領域を5ブロック＋退避領域1ブロック（ブロック1～ブロック6）とする。
- ④ 書き込み領域がなくなった場合には、消去可能フラグが立っている物理セクタが一番多いブロックの消去不可の物理セクタのデータを退避領域に移動して、移動元のブロックを消去する。そして、そのブロックを退避領域として使用し、今までの退避領域を書き込み領域とする（この一連の動作を退避動作と呼ぶ）。
- ⑤ 本体から送られて来るセクタのアドレスはAからPとする。

【0034】図25（a）において、セクタA～Gを書き込む。この場合には、同じ論理セクタがないので、そのまま、ブロック1から書き込む。（b）において、セクタA、D～Gを書き込む。この場合には、セクタA、D～Gと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。（c）において、セクタH～Nを書き込む。この場合には、同じ論理セクタがないので、そのまま書き込む。図26（d）において、セクタA、G～Lを書き込む。この場合、セクタA、G～Lと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。（e）において、セクタA、H～Jを書き込む。この場合、セクタA、H～Jと同一論理セクタがあるので、同一の論理セクタに消去可能フラグを立てたのち書き込む。

【0035】図27 (f-1) において、セクタA, I ~ Lを書き込むが、この場合には、書き込み領域がないので、退避動作を行った後書き込み処理を行う。すなわち、消去可能フラグが立っているデータが最も多いブロック (ブロック3) の消去可能でないデータを退避領域 (ブロック6) に移動して、移動元のブロックを消去する。そして、そのブロック (ブロック3) を退避領域として使用し、今までの退避領域を書き込み領域とする。ついで、(f-2) において、セクタA, I ~ Lを書き込み領域となったブロック6に書き込む。次に、(g) において、A, M, Nを書き込みを行うが、上記 (f) と同様書き込み領域がないので、まず、退避動作を行う。すなわち、消去可能フラグが立っているデータが最も多いブロック (ブロック5) の消去可能でないデータを退避領域 (ブロック3) に移動して、移動元のブロックを消去する。そして、そのブロック (ブロック5) を退避領域として使用し、今までの退避領域を書き込み領域とする。ここで、ブロック5が消去失敗により使用不可となったとすると、退避領域がなくなるので救済措置を行う。

【0036】そこで、図28 (h) に示すように、予備ブロック0を仮の退避領域として退避動作を行う。すなわち、図27 (g) において、消去可能でないデータが最も多いブロック (ブロック1) の消去可能でないデータを仮の退避領域 (ブロック0) に移動して、移動元のブロックを消去する。そして、そのブロック (ブロック1) を退避領域として使用する。ついで、(i) に示すように、予備ブロック (ブロック0) の退避動作を行う。すなわち、仮の退避領域として用いた予備領域 (ブロック0) の論理セクタB, Cをブロック3に移動する。これで、予備領域と退避領域が作成されたので、(j) において、論理セクタA, M, Nをブロック3に書き込む。次に、セクタA, G, H, O, Pを書き込むとするが、書き込み領域がないので、図29 (k-1) に示すように退避動作を行う。すなわち、消去可能なデータが最も多いブロック (ブロック4) の消去可能でないデータを退避領域 (ブロック1) に移動して、移動元のブロックを消去する。そして、そのブロック (ブロック4) を退避領域として使用する。ついで、(k-2) に示すように、セクタA, G, H, O, Pを書き込むが、セクタA, G, Hについては同じアドレスの論理セクタがあるので、消去フラグを立てたのちブロック1に書き込み、セクタO, Pについては、そのまま、書き込む。

【0037】次に、(l) において、セクタAを書き込むとするが、書き込み領域がないので、退避動作を行う。その結果、ブロック2が退避領域となり、ブロック2にあったセクタD, E, Fがブロック4に移動する。ここで、ブロック2が消去失敗で使用不可になったとすると、退避領域がなくなるので、救済措置を行う。すな

わち、消去可能なデータが最も多いブロック (ブロック3) のデータを仮の退避領域 (ブロック0) に移動して、移動元のブロックを消去する。そして、そのブロック (ブロック3) を退避領域として使用する。その結果、図30 (m) に示すようになる。なお、この状態で、電力不足などにより、処理に一旦ストップをかけたすると、処理を再開したとき、装置側は予備領域に消去可能なセクタがなく、データがあるので、空き領域作成中と判断できる。続いて、空き領域作成のための救済措置を行い、ブロック6のセクタI, Jを予備領域 (ブロック0) に移動し、ブロック6のセクタK, Lをブロック3に移動する。その結果図30 (n) に示すようになる。

【0038】ここで、退避領域のセクタを除いた空きセクタ数が、予備領域 (ブロック0) にあるデータの数以上なので、書き込みポイントを退避領域の後ろから検索し、空いている領域に、予備領域 (ブロック0) からデータを移動する。その際、図31 (o) に示すように、予備領域 (ブロック0) のセクタB, C, M, Nを移動した段階で、本体側が電力不足等により一旦処理にストップをかけたとする。この場合には、処理を再開した時に、装置側は予備領域に消去可能なセクタがあるので、空きブロック作成中の予備領域のデータ移動中と判断できる。続いて、図31 (p) に示すように、予備領域 (ブロック0) の論理セクタI, Jをブロック4に移動して処理を終了する。

【0039】図32は本実施例における空き領域作成処理を示すフローチャートであり、同図により、本実施例の処理を説明する。ステップS1において、予備領域に消去可能セクタが有るか否かを判別し、ある場合には、ステップS11に行く。ない場合には、ステップS2に行き、書き込みポイントを予備領域の先頭とし、ステップS3において予備領域にデータがあるか否かを判別する。予備領域にデータがある場合には、ステップS10に行き、ない場合にはステップS4に行き、退避領域を予備領域とする。ステップS5において、書き込み領域の各ブロックに消去可能セクタがない場合には、空き領域を作成できないので処理を放棄 (Abort) する。消去可能セクタがある場合には、ステップS6に行き、消去可能セクタの一番多いブロックを退避対象とし、ステップS7において、退避対象のブロックからデータを予備領域に移動する。

【0040】ステップS8において退避対象となるブロックを消去し、ステップS9において、退避対象のブロックを退避領域にする。ステップS10において、①予備領域のデータ数が②退避領域以外の空きセクタ数より多いか否かを判別し、①>②の場合には、ステップS5に戻り、上記処理を繰り返す。また、①≤②の場合には、ステップS11に行き、書き込みポイントを退避領域の次から書き込み方向に対して検索し、ステップS12に

において、空きセクタに予備領域からデータを移動する。ステップS13において、予備領域を消去して終了する。本実施例においては、上記のように空き領域を作成しているので、記憶媒体中に使用不可のブロックが発生しても、救済措置により空き領域を作成することができ、記憶装置して使用不可能となることを避けることができる。また、処理途中で、本体側の電力不足などにより処理が中断しても、どの段階で処理が中断したかを容易に判断することができ、処理中断における処理段階の判断を単純化することが可能となる。

【0041】C. 消去処理の効率化

フラッシュ・メモリは前記したように上書きをすることができず、消去した後でなければ書き込みを行うことができない。本実施例は、上記のようなフラッシュ・メモリにおいて、書き込み有無を示すフラグと、全空間消去処理を実行中であることを示す実行中フラグを設けることにより、全記憶空間を消去する際の消去処理を効率良く行うとともに、消去中に何らかの原因で処理が中断された場合においても、処理再開をすることができる実施例を示している。次に図33、図34により本実施例の処理を説明する。なお、本実施例においては、ブロック数は6個とし、各ブロックに書き込みフラグを書き込む領域を設け、ブロック6に全空間消去処理を実行中であることを示す実行中フラグを書き込む領域を設けている。図33(a)において、ブロック1、2、4の書き込み前に、ブロック1、2、4に書き込み有無フラグを書き込む。ついで、全空間をクリアする初期化処理に当たり、(b-1)に示すように、ブロック6に実行中フラグを書き込む。(b-2)において、書き込み有無フラグが書き込まれたブロック1を消去し、書き込む有無フラグを消す。

【0042】図33(b-3)において、書き込み有無フラグが書き込まれたブロック2を消去し、書き込む有無フラグを消す。ここで、ブロック2の消去後、何らかの原因により、ストップ・シーケンスに移り、その後立ち上がった場合でも、ブロック6に実行中フラグが書き込まれているので、直ちに初期化動作中と判断し、初期化動作に移ることができる。(b-4)において、ブロック1～3には書き込み有無フラグに書き込まれていないため、ブロック4を消去し、書き込む有無フラグを消す。(b-5)において、ブロック1～5には書き込み有無フラグに書き込まれていないため、ブロック6を消去し、実行中フラグを消す。

【0043】図35は本実施例の処理を示すフローチャートであり、同図により本実施例を説明する。ステップS1において、実行中フラグの書き込みがあるか否かを判別し、有る場合には、ステップS3に行き、ない場合にはステップS2において、最終処理ブロックに実行中フラグを書き込む。ステップS3において、処理対象をクリアし、ステップS4において、処理対象に書き込み

有無フラグがあるか否かを判別する。書き込み有無フラグがない場合には、ステップS6に行き、有る場合には、ステップS5において、処理対象を消去し、ステップS6において、処理対象を次に進める。ステップS7において、実行中フラグの書き込みがあるか否かを判別し、ある場合には、ステップS4に戻り上記処理を繰り返す。また、実行中フラグがない場合には、終了する。本実施例においては、上記のように書き込み有無フラグを設けているので、不要な消去を避けることができ、消去時間を短縮することができるとともに、媒体の寿命を延ばすことができる。また、初期化実行中であることを示す実行中フラグを設けているので、不意の中断に対応することができる。

【0044】D. 書き込み速度の向上、および、不良セクタ発生時の予備領域の割り当て

前記したように、フラッシュ・メモリは上書きをすることができず、消去した後でなければセクタの書き込みを行うことができない。このため、前記したように、書き込み時、同一のセクタがある場合には消去可能フラグを立て、適宜の時点で消去可能フラグが立っているセクタを消去する必要がある、書き込みの時間が掛かる。また、前記したように、フラッシュ・メモリは、消去回数に制限があり、所定回数以上消去を行うと消去が出来なくなる。このため、不良セクタが発生した場合には、予備の領域を割り当て救済措置を講ずる必要がある、以下に示す実施例は、上書きができず、セクタ単位で消去可能なメモリにおいて、上記した書き込み時のタイムラグをなくし、書き込み速度を向上するとともに、不良セクタが発生した場合に予備領域を割り当てる実施例を示している。図36は本実施例のシステム構成を示す図であり、図5に示したシステムに、EEPROM26が付加されており、その他の構成は図5に示したものと同一である。EEPROM26にはアドレス変換を行うデコード・テーブルが格納されており、上記EEPROM26によりアドレスのデコードが行われる。そして、フラッシュ・メモリの一部が不良になったとき、EEPROM26を書き換えて、救済措置を行う。なお、図36には、EEPROM26が例示されているが、デコード・テーブルを格納した記憶媒体としては、書き換えが可能な記憶媒体であればよく、その他フラッシュ・メモリ等を用いることができる。

【0045】(1) 実施例1

図37、図38は本実施例の構成を示す図であり、図37において、261はEEPROM26（もしくはフラッシュ・メモリ等の書き換え可能なROM）から構成されるセクタ変換部である。25はフラッシュ・メモリであり、フラッシュ・メモリは第1の領域A、第2の領域B、および予備領域Cが設けられ、第1の領域A、第2の領域Bには、それぞれ、1～4の4個のセクタが設けられている。図37において、データをセクタ1に書き

込む場合には、第1の領域Aと第2の領域Bを見て、既に第1の領域Aにデータが有る場合には、第2の領域Bにデータを書き込む。また、上記データを書き込んでいる間に、第1の領域Aのセクタ1-Aを消去する。その際、書き込みエラーが発生した場合には、予備領域Cを代替セクタとして割り当てる。すなわち、第1の領域Aのセクタ1-Aに書き込みエラーが発生した場合には、予備領域Cをセクタ1-Aの代替とするため、セクタ変換部261のROMを書き換え、図38に示すように、セクタ1-Aとして、予備領域Cを割り当てる。

【0046】以上のように、書き込みエラーが発生した場合、予備がある限り予備領域のセクタを割り当て、予備がなくなった場合には、データを全て図36に示したSRAM23等の外部の領域に退避し、セクタ変換部261のROMを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。本実施例においては、上記のように、書き込みと同時に消去を行っているため、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているので、不良セクタが発生した際の救済を行うことができる。

【0047】(2) 実施例2

図39、図40は本実施例の構成を示す図であり、図39において、261はEEPROM、もしくは、フラッシュ・メモリ等の書き換え可能なROMから構成されるセクタ変換部、25はフラッシュ・メモリであり、フラッシュ・メモリは第1の領域A、予備領域Cが設けられ、第1の領域Aには1~4の4個のセクタが設けられている。また、231は図36に示したSRAM23等のデータの一次記憶媒体である。図39において、データの書き込み時、一次媒体231にデータを転送している間に、書き込むセクタを調べ、書き込むべきセクタにデータがあった場合には、そのセクタを消去する。なお、転送が早く終了した場合には、消去終了まで待ち、また、転送が遅い場合には、転送終了後、データを書き込む。また、書き込みエラーが発生した場合には、セクタ変換部261のROMの書き換えをおこなって、図40に示すように(セクタ1-Aにエラーが発生した場合を示している)、予備領域Cにセクタ1-Aの書き込みを行う。なお、通常のセクタがエラーにならない限り、予備領域Cへの書き込みは行わない。本実施例においては、上記のように、データを一次記憶媒体231に転送している間に、書き込むべきセクタにデータがあった場合、消去を行っているため、第1の実施例と同様、書き込み時のタイムラグをなくすことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているの

で、不良セクタが発生した際の救済を行うことができる。

【0048】(3) 実施例3

図41、図42は本実施例の構成を示す図であり、図41において、261はEEPROM、もしくは、フラッシュ・メモリ等の書き換え可能なROMから構成されるセクタ変換部、25はフラッシュ・メモリであり、フラッシュ・メモリはブロック1、ブロック2、予備ブロック1、予備ブロック2が設けられ、各ブロックには、ブロック1、ブロック2には、セクタ1~3A、1~3B、…、1~3D、4~6A、4~6B、…、4~6Dが設けられている。そして、セクタ1~3A、1~3B、…、1~3Dは物理アドレスとしてはそれぞれ1セクタであり、ここに論理アドレス1、2または3のデータを書き込むことができる。すなわち、論理アドレス1、2または3のデータを書き込む領域がA~Dの4セクタ分設けられており、データを書き込む場合には、A~Dの内の空いている領域に書き込む。同様に、セクタ4~6A、4~6B、…、4~6Dも上記と同様であり、論理アドレス4、5または6のデータを書き込む場合には、A~Dの内の空いている領域に書き込む。以上のように、本実施例においては、 $n+1$ (この実施例においては $n=3$)セクタ単位でブロック化されており、書き込む論理アドレスに対して、1セクタの空きが設けられている。

【0049】図41において、データをセクタ1に書き込む際、セクタ1~3A、1~3B、…、1~3Dを見て、これらの中にセクタ1が既に存在する場合には、そのセクタを消去する。また、セクタには必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替ブロックとして予備ブロックを割り当てる。例えば、セクタ1~3Aが不良となった場合には、図42に示すように、予備ブロック1を代替ブロックとして割り当てる。そして、セクタ変換部261のROMを書き換えて、エラーのあったブロック1を予備ブロック1に割り当てる代替処理を行う。これにより予備ブロック1がブロック1となる。予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図36に示したSRAM23等の外部の領域に退避し、セクタ変換部261のROMを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。セクタ変換部261は、論理セクタから物理セクタへの変換時、上記したように、 $n+1$ セクタ単位でブロック化し、デコードするセクタ1に対して、書き込み可能な $n+1$ のセクタが存在するように構成する。そして、一部のセクタが不良となった場合には、上記セクタ変換部261のテーブルを書き換えるこ

とにより、書き込みブロックを移動して、常に $n+1$ のセクタのブロックを作成する。本実施例においては、上記のように、 $n+1$ セクタ単位でブロック化し、1セクタを空けておき、書き込みと同時に消去を行っているので、実施例1、2と同様、書き込み時のタイムラグをなくことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているため、不良セクタが発生した際の救済を行うことができる。

【0050】(4) 実施例4

図43、図44は本実施例の構成を示す図であり、図43において、261はEEPROM、もしくは、フラッシュ・メモリ等の書き換え可能なROMから構成されるセクタ変換部、25はフラッシュ・メモリであり、フラッシュ・メモリはブロック1、ブロック2、予備1A～1Dが設けられ、ブロック1、ブロック2には、それぞれ、セクタ1～3A、1～3B、…、1～3D、4～6A、4～6B、…、4～6Dが設けられている。また、本実施例においては、実施例3と同様、 $n+1$ （この実施例においては $n=3$ ）セクタ単位でブロック化し、1セクタを空けておく。図43において、データをセクタ1に書き込む際、セクタ1～3A、1～3B、…、1～3Dを見て、これらの中にセクタ1が既に存在する場合には、そのセクタを消去する。また、セクタには必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替セクタとして予備1A～1Dを割り当てる。例えば、セクタ1～3Aが不良となった場合には、図44に示すように、予備1A～1Dがブロック1となり、セクタ1～3Aに対して予備1Aを割り当て、セクタ変換部261のROMを書き換えて代替処理を行う。これにより予備セクタ1Aがセクタ1～3Aとなり、予備1B～1Dはブロック1の予備となる。

【0051】予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図36に示したSRAM23等の外部の領域に退避し、セクタ変換部261のROMを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。セクタ変換部261は、論理セクタから物理セクタへの変換時、上記したように、 $n+1$ セクタ単位でブロック化し、デコード時には、そのデコード・データよりアドレス変換するテーブルを設けセクタ1に対して書き込み可能な $n+1$ のセクタが存在するように構成する。そして、一部のセクタが不良となった場合には、上記セクタ変換部261のテーブルを書き換えることにより、常に $n+1$ のセクタのブロックを作成する。本実施例においては、上記のように、 $n+1$ セクタ単位でブロック化し、1セクタを空け

ておき、書き込みと同時に消去を行っているため、実施例1、2、3と同様、書き込み時のタイムラグをなくことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているため、不良セクタが発生した際の救済を行うことができる。

【0052】(5) 実施例5

図45、図46は本実施例の構成を示す図であり、図45、46において、図37、図38に示したものと同一のものには同一の符号が付されており、261はEEPROM、もしくは、フラッシュ・メモリ等の書き換え可能なROMから構成されるセクタ変換部、25はフラッシュ・メモリであり、フラッシュ・メモリはブロック1、ブロック2、予備ブロック1、予備ブロック2が設けられ、ブロック1、ブロック2、セクタ1～3A、1～3B、…、1～3D、4～6A、4～6B、…、4～6Dが設けられている。また、本実施例においては、実施例3と同様、 $n+1$ （この実施例においては $n=3$ ）セクタ単位でブロック化し、1セクタを空けておく。図45において、データをセクタ1に書き込む際、セクタ1～3A、1～3B、…、1～3Dを見て、これらの中にセクタ1が既に存在する場合には、そのセクタを消去する。また、必ず空いているセクタが存在するので、上記した消去と同時に、空いているセクタに書き込みを行う。書き込み時、書き込みエラーが発生した場合には、代替セクタとして予備セクタを割り当てる。例えば、セクタ1～3Aが不良となった場合には、予備ブロック1を割り当て、セクタ変換部261のROMを書き換えて、代替処理を行う。これにより、図46に示すように、予備ブロックのセクタ1、 n 、 m Aがセクタ1に置き変わり、予備ブロックにセクタ1が存在することとなり、元のブロックにはセクタ2、3がそのまま残る。

【0053】すなわち、この実施例においては、ブロックの構成は続きセクタでなくともよく、元のブロックの構成数を変更する。予備がある限り上記のような処理を行い、予備がなくなった場合には、データを全て図36に示したSRAM23等の外部の領域に退避し、セクタ変換部11のROMを再編成し、データの空間と予備の空間に分ける。また、その時、セクタは、システムの構成に応じて、昇順、または、降順で初期状態と同様に割り当てていくが、不良セクタは除外する。本実施例においては、上記のように、 $n+1$ セクタ単位でブロック化し、1セクタを空けておき、書き込みと同時に消去を行っているため、実施例1、2、3、4と同様、書き込み時のタイムラグをなくことができ、書き込み速度を向上させることができる。また、予備領域を設け、不良セクタが発生した場合に、予備領域に代替しているため、不良セクタが発生した際の救済を行うことができる。

【0054】図47～図49は上記実施例における処理を示すフローチャートであり、図36に示したシステム

における本実施例の処理を説明する。ステップS1において、本体からセクタデータの書き込み要求があると、ステップS2において、CPU22はコントローラLSI21からの通知によりデータの書き込み要求があることを認識する。なお、コントローラLSI21がCPU22に割り込みをかけて、データの書き込み要求を通知してもよい。ステップS3において、CPU22はEEPROM26のデータを読み込み、その書き込み場所を認識し、セクタデータが書かれているか否かをチェックするため、フラッシュ・メモリ25-1, ..., 25-5をアクセスし、ECCデータ、あるいは、フラッシュ・メモリに格納された管理情報等を参照する。そして、ステップS5において、フラッシュ・メモリにデータが書かれているか否かを判断し、データが書かれていない場合には、図48のステップS8に行く。また、データが書かれている場合には、ステップS6において、第2のセクタにデータを書き込むための情報を読み込むため、CPU22はEEPROM26をアクセスし、データを得る。ついで、ステップS7において、フラッシュ・メモリのデータを消去し（効率を上げるためエラーチェックは後で行う）、図48のステップS8に行く。

【0055】ステップS8において、本体よりデータをSRAM23にデータを転送し、終了を待つ。ステップS9において、フラッシュ・メモリにデータを書き込むため、CPU22はコントローラLSI21に対してデータの書き込みを通知し、コントローラLSI21はSRAM23よりフラッシュ・メモリにデータを書き込む。ついで、ステップS10において、CPU22はコントローラLSI21よりデータ書き込み終了の通知を受け、ステップS11において、データの書き込みエラーがあったか否かを判断する。データの書き込みエラーがなかった場合には終了し、書き込みエラーがあった場合には、ステップS12において、代替用セクタが有るか否かを判断し、代替用セクタがある場合には、ステップS13に行き、セクタアドレスを変更するため、EEPROM26をアクセスしてデータを変更し、ステップS9に戻る。また、代替用セクタがない場合には、図49のステップS14に行き、消去中のフラッシュ・メモリがあるか否かを判断し、消去中のフラッシュ・メモリがある場合には、ステップS15に行き、消去終了まで待つ、消去したセクタにデータを書き込み終了する。ステップS14において、消去中のフラッシュ・メモリがないと判断された場合には、ステップS16において、本体側にエラーを通知する。本体側では、ステップS17において、データを吸い上げEEPROM26を再編成するコマンドを送る。記憶装置側では、本体側からのコマンドに応じて、EEPROM26を再編成し、本体から送られる1論理セクタに対して、2セクタ以上の選択ができるようにする。これにより、少し容量は減るが記憶装置として使用可能となる。なお、上記処理に

おいては、SRAM23にデータを転送し、フラッシュ・メモリにデータを書き込む場合について説明したが、SRAM23を設けず、直接フラッシュ・メモリにデータを書き込むこともできる（この場合には、上記フローチャートのステップS8, S9は一処理となる）。

【0056】E. アドレス変換用デコード部の信頼性の向上

以下、本発明の請求項1の発明の実施例について説明する。論理アドレスを物理アドレスに変換するための変換テーブルとしてROMが使用されることがある。また、ROMの代わりに、フラッシュEEPROM、EEPROM等を使用することも可能であるが、通常はデコード部を可変にする必要がないため、使用されることが少ない。一方、フラッシュ・メモリは、その媒体の性質上、消去回数に制限があり、不良になると、不良になったアドレスが使用出来なくなる。そこで、前記したD. の実施例(1)～(5)に示すように、論理アドレスを物理アドレスに変換するための変換テーブルとして、フラッシュEEPROM、EEPROMなどの書き換え可能なデコード部を用い、不良のセクタが発生した場合、デコード部を書き換え、不良アドレスがないように見せかける方法が用いられる。しかしながら、変換テーブルとして、フラッシュEEPROM、EEPROMなどを用いた場合、これらのフラッシュEEPROM、EEPROMなどにも、消去回数に制限があり、所定回数以上消去を行うと不良が発生する。本実施例は、上記のように、デコード部としてフラッシュEEPROM、EEPROMなどの書き換え可能な記憶媒体が使用されている場合において、デコード部を2段化とすることにより、デコード部の寿命を延ばし、アドレス変換用デコード部の信頼性を向上させる実施例を示している。

【0057】図50は本実施例のデコード部の構成を示す図であり、同図(a)は通常の場合、(b)はセクタが不良になった場合、(c)はデコード部が不良になった場合を示しており、301は1次デコード部、302は2次デコード部、303はフラッシュ・メモリのセクタを示している。同図において、フラッシュ・メモリのセクタが正常な場合には、(a)に示すように、1次デコード部301がセクタ0のアドレスのデコード値としてアドレス「0000h」を出力し、2次デコード部302が「0000h」のデコード値として物理アドレス「5555h」を出力し、フラッシュ・メモリのセクタ0の物理アドレスを指定している。ここで、フラッシュ・メモリの消去回数が制限値を越えてセクタが不良となり、セクタ0の物理アドレスをアドレス8888hの予備のセクタに切り替える場合には、同図(b)に示すように、2次デコード部302を書き換えて、「0000h」のデコード値を「5555h」から「8888h」に切り替える。これにより、セクタ0として物理アドレス「8888h」が割り当てられる。上記のようにし

て、2次デコード部(もしくは1次デコード部)を何度か書き換えることにより書き換え制限値以上になり、2次デコード部に不良になると、同図(c)に示すように、一次デコード部301を書き換えてそのデコード値をずらし、2次デコード部302がデコード値として「8888h」を出力する「2222h」をデコード値として出力するように書き換える。これにより、2次デコード部302が不良になっても、物理アドレス「8888h」をデコード値として出力することができる。

【0058】以上のように、デコード部を2段化し、一方のデコード部が不良になった場合に、他方のデコード部を書き換えることにより、デコード部の寿命を2乗化することができる。例えば、セクタが不良になるまでの回数をL回(書き換え制限値)、2次デコード部が不良になるまでの回数をM回(書き換え制限値)、1次デコード部が不良になるまでの回数をN回(書き換え制限値)とすると、アドレスがエラーになるまでの回数は $N \times M \times L$ 回となる。基本的に書き換え制限値はフラッシュEEPROMで10万~100万回、EEPROMで1万回程度といわれているので、事実上、デコード部を2段化するだけで、信頼性を十分確保することができる。

【0059】F. 書き込み時間の推定処理
フラッシュ・メモリにおいては、データの書き込みの際、退避処理等を行う必要があり、通常の半導体メモリの場合より書き込みに時間がかかる。本実施例は、フラッシュ・メモリへの書き込み時間の推定を行い、消費電力の推定、異常の検出等を行う実施例を示している。図51に本実施例のシステム全体の概略構成を示す。1は記憶装置、1aは上書きすることができず、また、セクタ、ブロック等の所定の単位でしかデータを消去することができない、例えばフラッシュ・メモリ等から構成される記憶領域、1bはコントローラ、2は本体処理装置である。次に本実施例について説明する。今、記憶装置が図52に示すような状態になっている場合に、3セクタのデータを書き込む場合を考える。まず、図51の本体よりプロセッサ22に書き込むセクタ数を送る。書き込むセクタ数がプロセッサ22に送られると、プロセッサは書き込むセクタ数により書き込み時間を求める。ここで、図52のように書き込むブロックが無い場合には、退避動作が行われるので、書き込み時間 t は次のようにして求められる。

$$t = [1 \text{セクタの書き込みに掛かる時間}] \times 3 + [\text{データの退避時間}] \text{ (秒)}$$

プロセッサ22は上記のようにして書き込み時間を計算し、本体へ返す。本体は、上記書き込み時間を元に、次の式により書き込み動作時の消費電力 $W1$ を求める。

$$W1 = t \div 3600 \times [\text{書き込み動作の消費電力}] \text{ (W)}$$

また、本体はその電源部から残りの電力 $W2$ を読み取

り、得られた書き込み電力と比較し、 $W2 > W1$ ならデータの書き込みを行う。一方、本体は上記のようにして得た書き込み時間 t と実際の書き込み時間を比較し、上記書き込み時間 t を越えても書き込みが終了しない場合には、記憶装置の異常と判断し、例えば、ユーザに通知したり、書き込み処理をストップ等の措置を行う。

【0060】図53は本実施例における本体の処理を示すフローチャートであり、同図により本実施例を説明する。ステップS1において、これから書き込むデータのサイズを記憶装置に送り、書き込みにかかる時間を得る。ステップS2において、書き込みにかかる時間と、時間当たりの消費電力をかけて書き込みにかかる電力を得る。ステップS3において、残り電力と書き込みで使用する電力を比較し、残り電力の方が少ない場合には、異常処理を行う。また、残り電力の方が多い場合には、ステップS4に行きデータを書き込み、ステップS5において、実際の書き込み時間が予想した時間より多いか否かを判別し、多い場合には、異常処理を行う。また、書き込み時間が予想した時間以内の場合には、ステップS6に行き、書き込み終了か否かを判別し、書き込み終了でない場合にはステップS5に戻り、書き込み終了の場合には、終了する。

【0061】以上のように、本実施例においては、書き込み時間を求めることにより、書き込みに要する電力を推定しているので、残り電力量で書き込みができるかどうかの判断を行うことができる。このため、電池駆動等のシステムにおいて、書き込み中に電力がなくなり書き込みが途中で中断することを避けることができ、システムの信頼性を向上させることができる。また、実際の書き込み時間と推定した書き込み時間を比較することにより、記憶装置の異常を検出することができ、また、書き込み時間を表示するなどしてユーザに知らせることにより、ユーザが記憶装置の異常を判断することが可能となる。

【0062】G. フラグ判定処理における信頼性の向上
フラグの判定は、通常、1ビットを1機能に割り当て判定している場合が多い。しかしながら、フラッシュ・メモリは過剰消去により消去不能になるという、致命的不良原因を構造的にもっており、これが発生したセルはハイレベルのままで、ローレベルに戻らない状態に陥る。このため、フラッシュ・メモリ上のフラグを1ビット1機能に対応させた場合には、そのビットが不良になったとき、フラグの有効性がなくなってしまう。本実施例は、前記図51に示したようにフラッシュ・メモリ上の消去フラグ、不良フラグ、パリティ等に対して、フラグを複数用意し論理積によりフラグ判定することにより、上記したフラッシュ・メモリの構造的欠点に対処し、フラグ判定の信頼性を向上させた実施例を示している。

【0063】(1) 実施例1

図54は実施例1を示す図であり、同図(a)はフラグレジスタ、(b)は判定のための論理積回路、(c)は真理値表を示している。本実施例においては、同図(a)に示すように、一つの機能に対してフラグb0、b4を用意してフラグレジスタに格納する。そして、フラグ判定を行う場合には、(b)に示すように、これらのフラグの論理積を求めることにより、最終的な判定結果を得る。上記のように判定しているため、同図(c)の行番号2のようにビットb0がハイレベルで固定になっても、その論理積結果としては、正しい結果を得ることができる。同様に、行番号3のように、ビットb4がハイレベルで固定になっても、その論理積結果としては、正しい結果を得ることができる。

【0064】(2) 実施例2

図55は実施例2を示す図であり、同図(a)は第1のフラグレジスタ、(b)は第2のフラグレジスタ、(c)はフラグ判定を行う論理積回路を示している。本実施例は同一機能に対するフラグを別々のフラグレジスタに格納するようにしたものであり、同図(a)に示す第1のフラグレジスタのビットb0にフラグを格納し、同図(b)に示す第2のフラグレジスタのビットb0とビットb2に第1のフラグレジスタに格納したフラグと同一の機能に対応するフラグを格納している。そして、フラグを判定する場合には、同図(c)に示す論理積回路により上記フラグの論理積を求める。本実施例においても、実施例1と同様、フラグの一部がハイレベルで固定しても、正しい判定結果を得ることができる。以上のように、本実施例においては、フラグの判定を複数ビットで行うようにしているため、全てのビットが不良にならない限り、正しい判定値を常に出力することができる。フラグ判定の信頼性を向上することができる。

【0065】H. 管理テーブルの節約

本実施例は記憶装置が複数チップで構成され、各チップ内に複数のブロックが設けられたフラッシュ・メモリ・システムにおいて、管理テーブルを簡便化するとともに、各ブロックを平均的に使用することができる実施例を示している。以上示した実施例においては、データを書き込むチップ、ブロックに制約がなく、データは全ブロックのセクタに書き込むことができたため、全領域を管理するテーブルを設ける必要があったが、本実施例においては、各チップ内に書き込むデータを固定とし、チップ内において書き込むブロックを移動して書き込んでいくことにより、全領域を管理することなくデータの書き込みができるようにし、フラッシュ・メモリの管理を簡便化するとともに、各ブロックの使用を平均化している。また、本実施例においては、チップ内に書き込まれるデータをチップの全容量より一定量少なくし、この一定量のブロックをワークブロックと不良ブロック発生時の救済ブロックに当てている。そして、ブロック内のデータの並びは固定とし、データを書き換える場合、すで

に旧ブロックに有効なデータが存在している場合には、そのデータを前記した図5のSRAM23に退避して、新データと一緒に空いている領域に書き込み、書き込みが成功したら旧データのブロックを消去するようにしている。

【0066】次に本実施例を説明する。図56は本実施例におけるチップ、ブロック、セクタ内構成を示す図であり、同図に示すように、チップは2Mbyteであり、チップ内は1ブロック4KbyteのNo. 000～No. 511の512個のブロックに分割され、各ブロック内には、1セクタ528byteのNo. 00～No. 07の8個のセクタが設けられている。なお、同図のブロックNo. およびセクタNo. は物理アドレスを示している。さらに、セクタ内には、同図に示すように、256byteの実データ領域、Long命令用のECC領域、不良データフラグ、不良ブロックフラグ、ブロックアドレス、256byteの実データ用領域ECC領域が設けられている。ブロック・アドレスは、SRAM23に格納されているブロック・ポインタのアドレス値を示し、ブロック・アドレスには、実データなしと判断したセクタも書き込むこととしている。このようにすることにより、活線抜き差し時、セクタアドレス数が多い物が正常データと判断することができる。なお、セクタアドレス数が等しい場合はECC・チェックサムで正誤で判断する。不良ブロックフラグは、ブロックのコンディションを示しており、FFhで正常ブロック、≠FFhで不良ブロックとしている。不良データフラグは、データのコンディションを示しており、FFhで正常データ、≠FFhで不良データとしている。Long命令用のECC領域は最大4byteである。ECC領域は、実データのコンディションを示しており、この領域の値によりデータの訂正および誤り検出を行う。

【0067】図57～図61は本実施例における書き込み処理を示す図であり、同図により本実施例を説明する。本実施例においては、チップがNo. 00～No. 04の5個で、各チップにワークブロックを4個設けた場合の書き込みおよび管理方式を示している。図57は全ブロックを消去した状態を示しており、同図に示すように、各チップにはワークブロックがwork01～work04の4個設けられており、データは上記ワークブロックwork01～work04に書き込まれる。書き込みはセクタ単位で可能であるが、セクタ005の書き込み後にセクタ004を書き込むなど、上位のセクタの書き込み後には下位のセクタの書き込みはできない。また、チップを越えてデータが移動することではなく、ブロック内のセクタの順番は不動である。さらに、例えば、論理ブロックアドレス00にセクタ000～005のみを書き込む場合のように、書き込みデータが8セクタ全てを満たさない場合には、セクタ000～005のデータを書き込み後、セクタ006、007にブロックアドレスのみを書

き込む。なお、この場合、1ブロックが8セクタであり、また、チップが5個であるので、論理アドレス n のデータは、(論理アドレス $n \div 8$)の商をさらに5で割ったときに得られた余りのチップに書き込まれる。例えば、 $n=121$ の場合には、 $121 \div 8$ は15で余り1であり、 $15 \div 5$ は3で余り0なので、No. 00のチップに書き込まれる。

【0068】本実施例におけるデータの書き込み処理は次のように行われる。まず、図57に示す全ブロックを消去した状態からセクタNo. 000~039(論理アドレス)の40セクタを連続書き込みすると、図58に示すように各チップのwork01にデータが書き込まれる。すなわち、最初に書き込まれるセクタNo. 000~007は論理ブロックアドレスが00なので、本体からSRAM23にセクタNo. 000~007のデータを転送して、チップNo. 00の書き込みポインタが指すwork01からデータを書き込む。なお、論理ブロックアドレスはチップ単位で管理される。そして、この場合には論理ブロックアドレス00が過去の存在しないので、チップNo. 00のwork01はwork04の次の空き領域に移動する。8セクタ分の書き込みが終了したら、書き込みポインタをチップNo. 01のwork01に移動し、上記と同様にセクタNo. 008~015のデータをwork01に書き込む。

【0069】以上のようにして、書き込みがチップNo. 04まで完了したら、書き込みポインタをチップNo. 00のwork02に移動する。次に、図58の状態から、図59に示すようにセクタNo. 120~191(論理アドレス)の72セクタの連続書き込みをする。上記と同様、本体からSRAM23にセクタNo. 120~127のデータを転送して、セクタNo. 120~127は論理ブロックアドレスが03なので、チップNo. 00の書き込みポインタが指すwork02からデータを書き込む。そして、上記と同様、チップNo. 00のwork02、work03はwork01の次の空き領域に移動する。以下同様に、セクタNo. 128~191のデータをチップNo. 00~No. 04に書き込む。ここで、図59の状態から、セクタNo. 002~003の2セクタを連続書き込みすると、図60に示すようになる。まず、SRAM23にセクタNo. 002~003のデータを転送する。この場合、書き込みデータがセクタNo. 002~003なので、論理ブロックアドレス00が指定され、これは過去に存在するので、旧データの退避が必要と判断される。また、書き込みデータがセクタNo. 002~003で、論理ブロックアドレス00の最初から書き込まれないので、旧論理ブロックアドレス00のデータをSRAM23に退避する。なお、書き込みデータが論理ブロックアドレスの先頭より2セクタ分のときには、書き込みデータを先に処理した後に旧論理ブロックアドレスのデータ

の退避動作を行う。

【0070】次に、SRAM23に退避した旧データ000, 001, 004, 005, 006, 007と新データ002, 003を書き込みポインタが指すチップNo. 00のwork04に書き込む。その際、書き込みは000, 001, 002, 003, 004, 005, 006, 007の順番に書き込む。ついで、論理ブロックアドレス00がすでに存在しているので、旧論理ブロックアドレスの消去を行う。また、work04を旧論理ブロックアドレス00に移動する。上記のように2セクタ連続書き込みを行ったのち、電源がOFFになり、再びONにすると、図61に示すように、ワークブロックは使われているブロックの次より指定される。また、上記のようにワークブロックを指定していき、チップの最後まで検索してもワークブロックを全部指定できなかった場合には、チップの最初に戻り、ワークブロックを指定する。上記の処理において、ブロック消去エラーが発生した場合には、不良フラグに00hを設定し、そのブロックは書き込みおよび読み出し不可とする。すなわち、ワークブロックを一つ潰すことにより対処し、ワークブロックがなくなったら、そのシステムは書き込み不可とする。また、データ退避途中にエラーが発生した場合には、不良データフラグを立てたのちにデータを書き込む。なお、読み出し動作ではそのブロックにエラーが検出されても、そのブロックを不良ブロックとしては扱わない。データ書き込みエラーが発生した場合には、そのブロックを不良ブロックとする。不良フラグの立て方は、一度書かれているデータを退避し、不良ブロックフラグのみを立てて、再度書き込む。不良ブロックフラグも通常書き込みと同様、該当ブロック全てのセクタに対して立てる。データ消去途中でエラーが検出された場合には、そのブロックは不良ブロックとする。不良ブロックフラグの立て方は、不良ブロックフラグのみを立てるのではなく、実データを含め全てALL(00)を書き込む。不良処理は該当ブロック全てのセクタに対して行う。

【0071】以上のように、本実施例においては、各チップ内に書き込むデータを固定とし、チップ内において書き込むブロックを移動して書き込んでいくので、全領域を管理する必要がない。このため、データを各チップ内でのみ管理すればよく、管理テーブルを節約することができ、また、書き込み動作の高速化を図ることができる。また、ワークブロックにデータを書き込み、ワークブロックをチップ内で移動させるように構成したので、各ブロックを平均的に使用することができる。

【0072】

【発明の効果】以上説明したように、本発明においては、デコーダを2段設け、記憶装置の記憶領域の一部が破壊したとき、もしくは、デコーダの一部が破壊したとき、2段のデコーダのいずれか一方、もしくは、両方を

書き換えることにより、破壊した部分へのデコードが行われなようにしたので、デコーダとして、EEPROM、フラッシュメモリ等のその一部が破壊する可能性のある記憶媒体を用いた場合においても、アドレスがエラーとなる確率を著しく減少させることができ、信頼性を確保することができる。

【図面の簡単な説明】

- 【図1】本発明の全体の概略構成を示す原理図である。
- 【図2】本発明の概要を示す原理図（1）である。
- 【図3】本発明の概要を示す原理図（2）である。
- 【図4】本発明の概要を示す原理図（3）である。
- 【図5】本発明の前提となる記憶装置の構成を示すブロック図である。
- 【図6】記憶装置におけるSRAMの内容を示す図である。
- 【図7】記憶装置におけるフラッシュ・メモリの内容を示す図である。
- 【図8】記憶装置におけるフラッシュ・メモリの内容を示す図（続き）である。
- 【図9】記憶装置における書き込み時の処理を示すフローチャートである。
- 【図10】記憶装置における書き込み時の処理を示すフローチャート（続き）である。
- 【図11】記憶装置における書き込み時の処理を示すフローチャート（続き）である。
- 【図12】記憶装置における書き込み時の処理を示すフローチャート（続き）である。
- 【図13】記憶装置における書き込み時の処理を示すフローチャート（続き）である。
- 【図14】消去回数のバラツキを抑える実施例を示す図である。
- 【図15】消去回数のバラツキを抑える実施例を示す図（続き）である。
- 【図16】消去回数のバラツキを抑える実施例を示す図（続き）である。
- 【図17】消去回数のバラツキを抑える実施例を示す図（続き）である。
- 【図18】消去回数のバラツキを抑える実施例を示す図（続き）である。
- 【図19】消去回数のバラツキを抑える実施例の処理を示すフローチャートである。
- 【図20】消去可能データをなくす処理の実施例を示す図である。
- 【図21】消去可能データをなくす処理の実施例を示す図（続き）である。
- 【図22】消去可能データをなくす処理の実施例を示す図（続き）である。
- 【図23】消去可能データをなくす処理の実施例を示す図（続き）である。
- 【図24】消去可能データをなくす処理の実施例を示す

フローチャートである。

【図25】空き領域の作成処理の実施例を示す図である。

【図26】空き領域の作成処理の実施例を示す図（続き）である。

【図27】空き領域の作成処理の実施例を示す図（続き）である。

【図28】空き領域の作成処理の実施例を示す図（続き）である。

【図29】空き領域の作成処理の実施例を示す図（続き）である。

【図30】空き領域の作成処理の実施例を示す図（続き）である。

【図31】空き領域の作成処理の実施例を示す図（続き）である。

【図32】空き領域の作成処理の実施例のフローチャートである。

【図33】消去処理の実施例を示す図である。

【図34】消去処理の実施例を示す図（続き）である。

【図35】消去処理の実施例のフローチャートである。

【図36】書き込み速度の向上を図る実施例のシステム構成を示す図である。

【図37】書き込み速度の向上を図る第1の実施例を示す図である。

【図38】書き込み速度の向上を図る第1の実施例を示す図（続き）である。

【図39】書き込み速度の向上を図る第2の実施例を示す図である。

【図40】書き込み速度の向上を図る第2の実施例を示す図（続き）である。

【図41】書き込み速度の向上を図る第3の実施例を示す図である。

【図42】書き込み速度の向上を図る第3の実施例を示す図（続き）である。

【図43】書き込み速度の向上を図る第4の実施例を示す図である。

【図44】書き込み速度の向上を図る第4の実施例を示す図（続き）である。

【図45】書き込み速度の向上を図る第5の実施例を示す図である。

【図46】書き込み速度の向上を図る第5の実施例を示す図（続き）である。

【図47】書き込み速度の向上を図る実施例のフローチャートである。

【図48】書き込み速度の向上を図る実施例のフローチャート（続き）である。

【図49】書き込み速度の向上を図る実施例のフローチャート（続き）である。

【図50】アドレス変換用デコード部を2段とした実施例を示す図である。

【図51】書き込み時間の推定処理を行うシステム全体の概略構成を示す図である。

【図52】書き込み時間推定処理の実施例における記憶領域の状態を示す図である。

【図53】書き込み時間推定処理の実施例のフローチャートである。

【図54】フラグ判定処理の第1の実施例を示す図である。

【図55】フラグ判定処理の第2の実施例を示す図である。

【図56】管理テーブルを節約する実施例における記憶領域の構成を示す図である。

【図57】管理テーブルを節約する実施例を示す図である。

【図58】管理テーブルを節約する実施例を示す図（続き）である。

【図59】管理テーブルを節約する実施例を示す図（続き）である。

【図60】管理テーブルを節約する実施例を示す図（続

き）である。

【図61】管理テーブルを節約する実施例を示す図（続き）である。

【符号の説明】

1, 20

1a

1b

1c

1d, 1e

2

21

22

23

24

25-1~25-5

26

261

301

302

記憶装置

記憶領域

一次記憶媒体

制御手段

デコード・テーブル

本体処理装置

コントローラLSI

プロセッサ

SRAM

クロック発振器

フラッシュ・メモリ

EEPROM

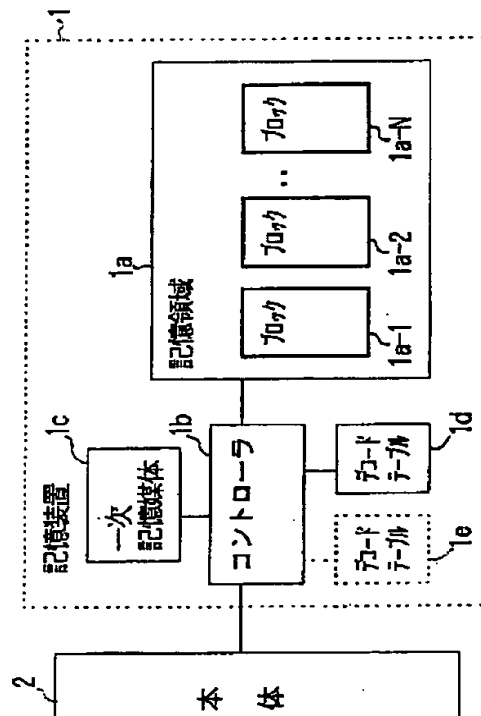
セクタ変換部

1次デコード部

2次デコード部

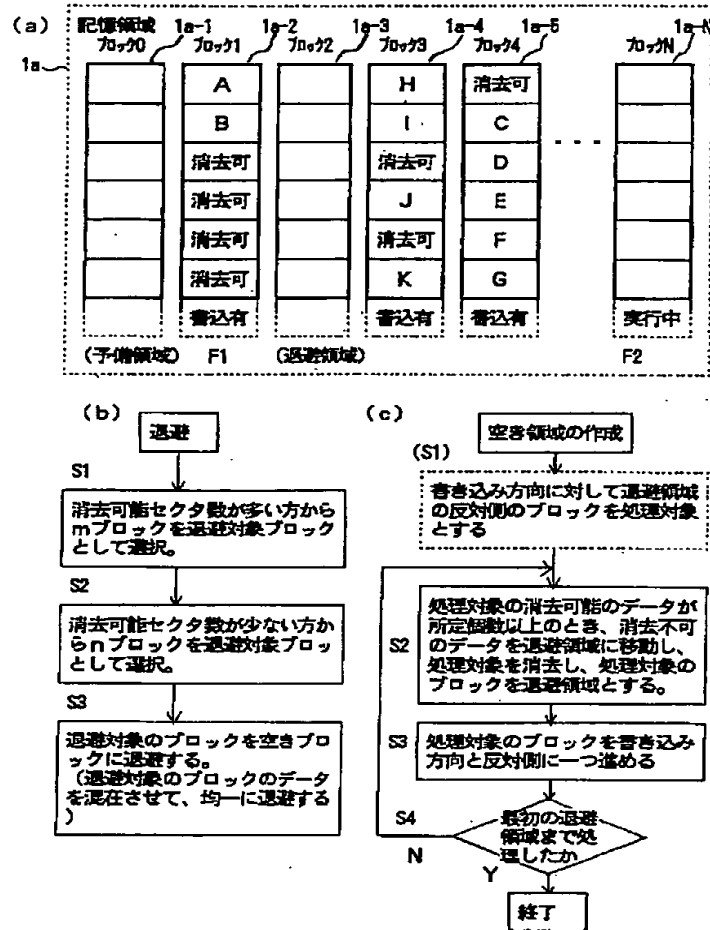
【図1】

本発明の全体の概略構成を示す原理図



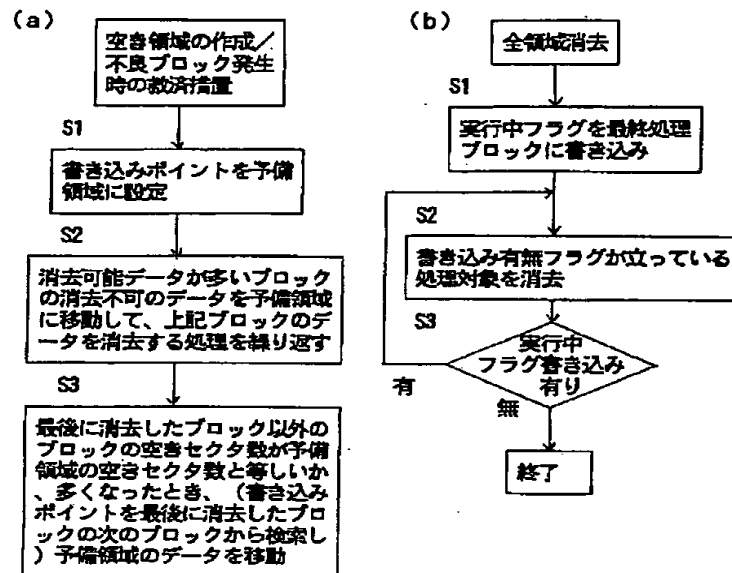
【図2】

本発明の原理を示す原理図 (1)



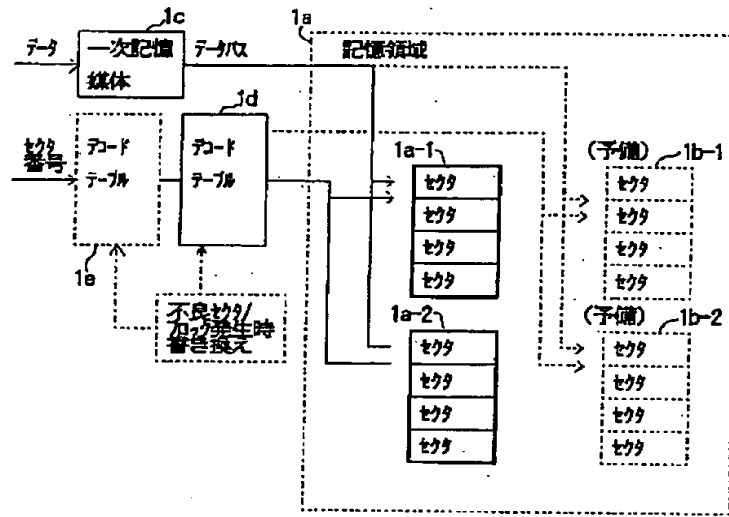
【図3】

本発明の原理を示す原理図(2)



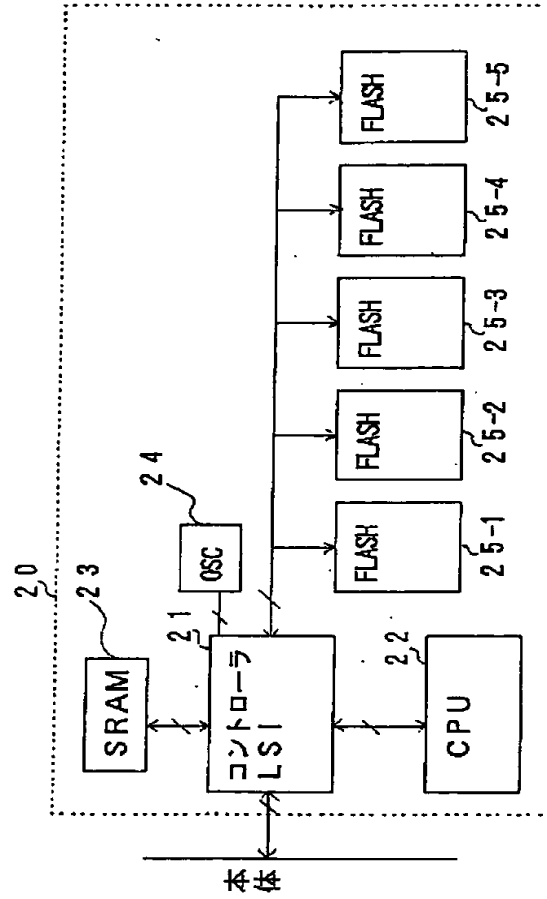
【図4】

本発明の原理を示す原理図（3）



【図5】

本発明の前提となる記憶装置の構成を示すブロック図



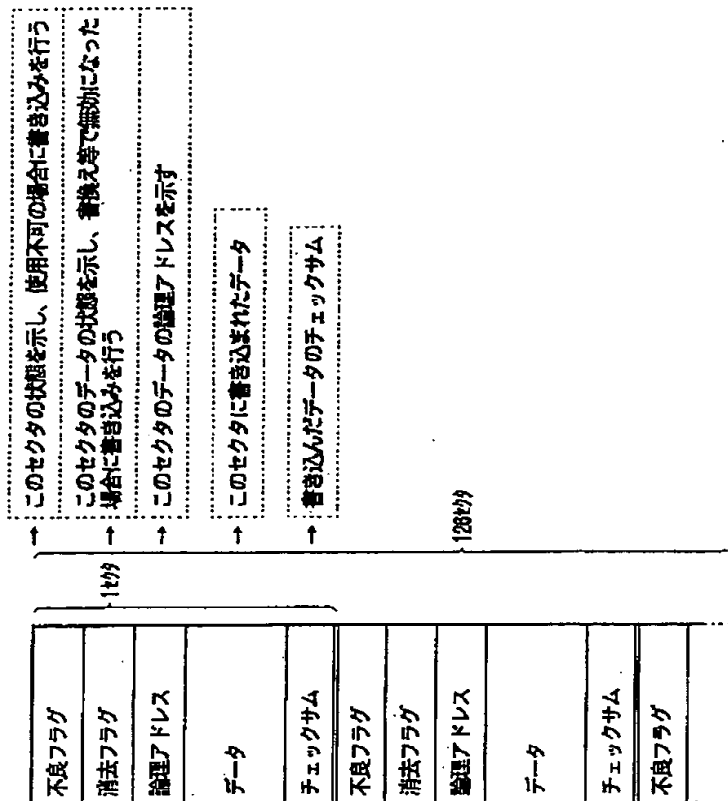
【図6】

記憶装置におけるSRAMの内容を示す図

消去回数テーブル	→	各ブロックの消去回数を保持
消去可能セクタ数テーブル	→	各ブロックの消去可能セクタフラグの立っている総数値を保持
書き込みポインタ	→	FLASH に書き込みを開始するChip No., Block No., セクタNo. を保持
WORK-BLOCK-No.	→	現在のWORK-BLOCK-No. を示し, ChipNo., Block No. を保持
整理ポインタ	→	現在、整理をしているChipNo., Block No., セクタNo. を保持
書換え有無	→	現在書き込みするデータが新規かどうかを示す
書き込み回数	→	整理時の本体書き込み回数を管理する
退避カウンタ	→	整理時の退避動作のセクタ数を管理する
チップ数	→	カードの搭載Flash-Chip数を保持する
セクタマップテーブル	→	論理アドレスの変換用にChipNo., Block No., セクタNo. を保持

【図7】

記憶装置におけるフラッシュ・メモリの内容を示す図



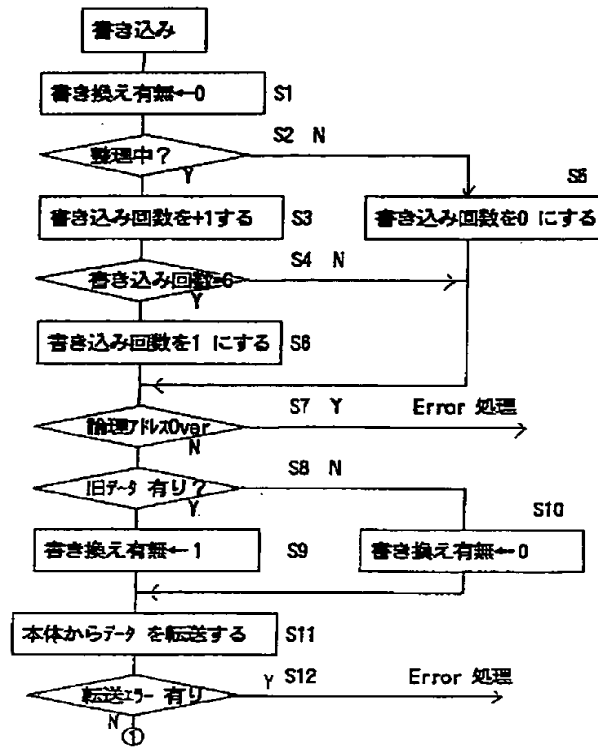
【図8】

記憶装置におけるフラッシュ・メモリの内容を示す図 (続き)

チェックサム	→	整理対象ブロックの中の不良セクタ
不良セクタメモリ	→	整理対象ブロックの消去回数
整理元消去回数	→	消去回数
消去回数	→	データを回避してのChip No., Block No.
回避ブロックNo	→	整理対象ブロックの消去を開始する時にたてる
消去start	→	整理対象ブロックの消去を終了するときにたてる
消去end	→	All Erase 時に消去対象として指定した時にたてる
All Erase 対象	→	ブロック内にデータがあるか示し、データがある場合にたてる
空きブロック	→	このブロックの状態を示し、使用不可になったときにたてる
ブロックステータス	→	

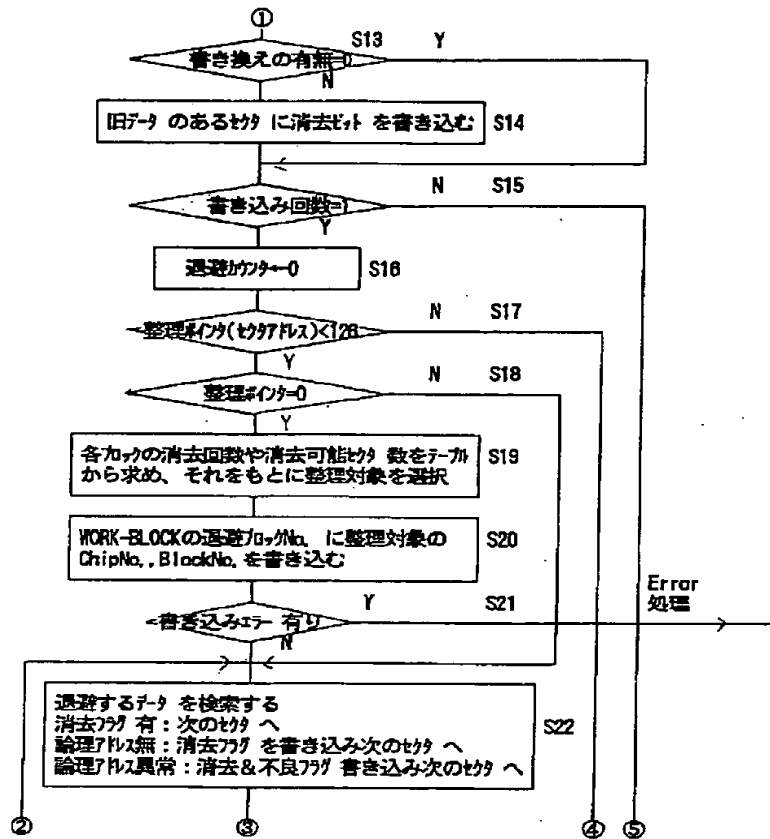
【図9】

記憶装置における書き込み時の処理を示すフローチャート



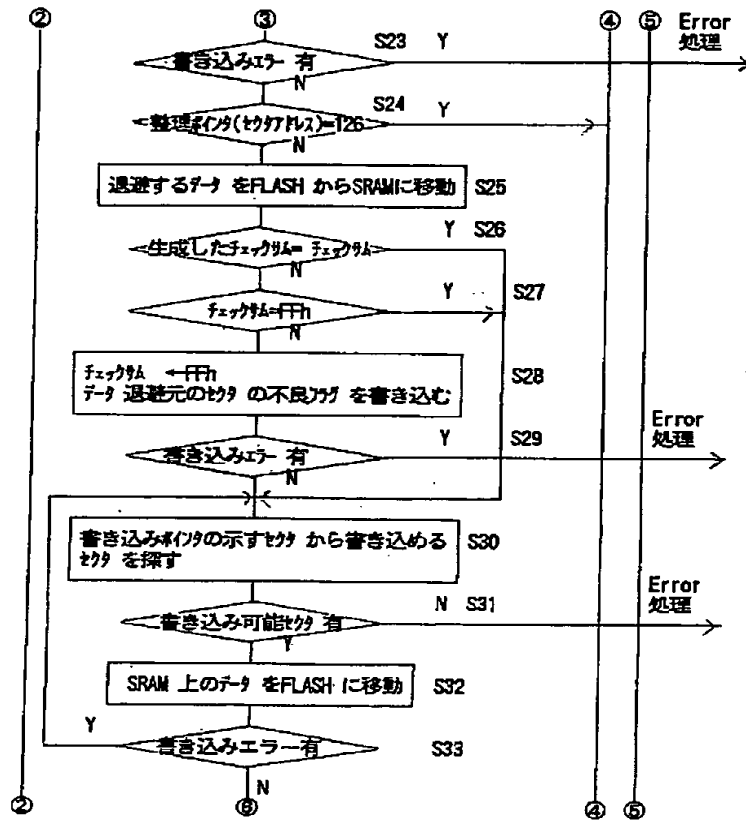
【図10】

記憶装置における書き込み時の処理を示すフローチャート（続き）



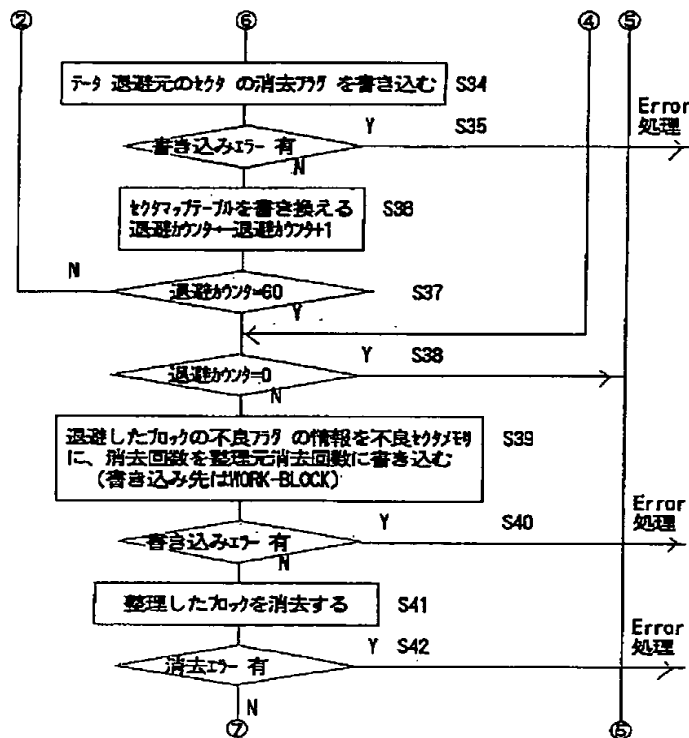
【図11】

記憶装置における書き込み時の処理を示すフローチャート（続き）



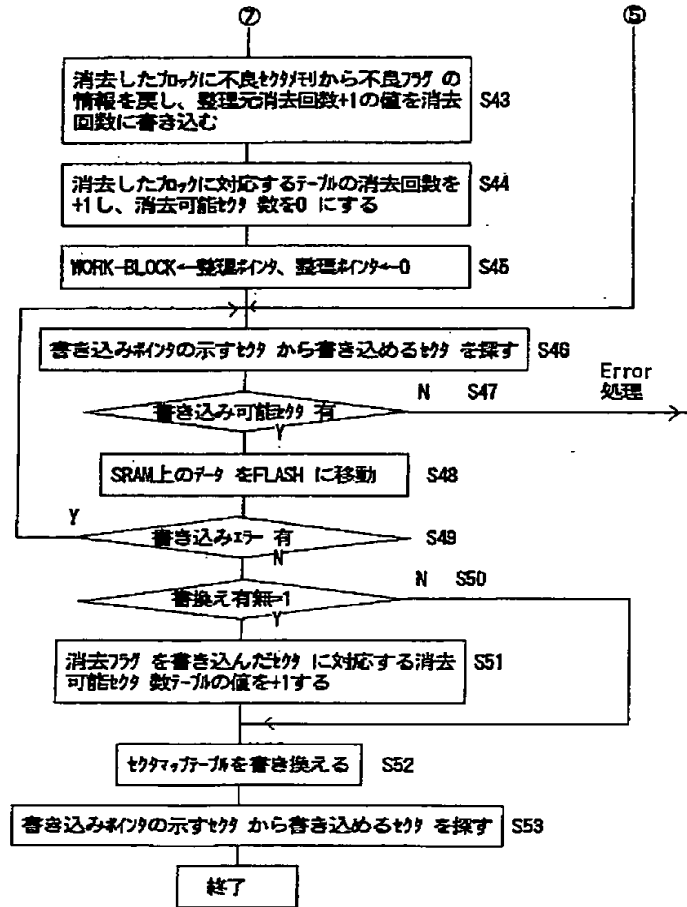
【図12】

記憶装置における書き込み時の処理を示すフローチャート（続き）



【図13】

記憶装置における書き込み時の処理を示すフローチャート（続き）



【図14】

消去回数のバラツキを抑える実施例を示す図

(a) A, B, C, D, E, F, G, H, Iの書き込み

フロッカ1	フロッカ2	フロッカ3	フロッカ4	フロッカ5	フロッカ6
A	G				
B	H				
C	I				
D					
E					
F					

(b) C, Dの書き込み

フロッカ1	フロッカ2	フロッカ3	フロッカ4	フロッカ5	フロッカ6
A	G				
B	H				
消去可	I				
消去可	C				
E	D				
F					

【図15】

消去回数のバラツキを抑える実施例を示す図 (続 き)

(c) J, K, L, M, N, O の書き込み

加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
A	G	K			
B	H	L			
消去可	I	M			
消去可	C	N			
E	D	O			
F	J				

(d) H, I, J, K, L, M, N の書き込み

加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
A	G	消去可	I		
B	消去可	消去可	J		
消去可	消去可	消去可	K		
消去可	C	消去可	L		
E	D	O	M		
F	消去可	H	N		

【図26】

空き領域の作成処理の実施例を示す図 (続 き)

(d) A, G ~ L の書き込み

加ッ0	加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
	消去可	消去可	消去可	N	K	
	B	消去可	消去可	A	L	
	C	D	消去可	G		
	消去可	E	消去可	H		
	消去可	F	消去可	I		
	消去可	消去可	M	J		

(予 備 領 域) (退 避 領 域)

(e) A, H ~ J の書き込み

加ッ0	加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
	消去可	消去可	消去可	N	K	
	B	消去可	消去可	消去可	L	
	C	D	消去可	G	A	
	消去可	E	消去可	消去可	H	
	消去可	F	消去可	消去可	I	
	消去可	消去可	M	消去可	J	

(予 備 領 域) (退 避 領 域)

【図16】

消去回数のバラツキを抑える実施例を示す図（続き）

(e-1) C, Dの書き込みのための選避

加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
A	G			O	H
B	消去可			I	J
消去可	消去可			K	L
消去可	C			M	N
E	D				
F	消去可				

↓

(e-2) C, Dの書き込み

加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
A	G			O	H
B	消去可			I	J
消去可	消去可			K	L
消去可	消去可			M	N
E	消去可			C	
F	消去可			D	

【図17】

消去回数のバラツキを抑える実施例を示す図（続き）

(f-1) H, I, Jの書き込み（J書き込み時点で空きブロックなし）

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A	G			O	消去可
B	消去可			消去可	J
消去可	消去可			K	L
消去可	消去可			M	N
E	消去可			C	H
F	消去可			D	I

↓

(f-2) 退避

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A		G	J	O	
B		L	N	消去可	
消去可		H	I	K	
消去可				M	
E				C	
F				D	

↓

(f-3) Jの書き込み

ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
A		G	消去可	O	
B		L	N	消去可	
消去可		H	I	K	
消去可		J		M	
E				C	
F				D	

【図18】

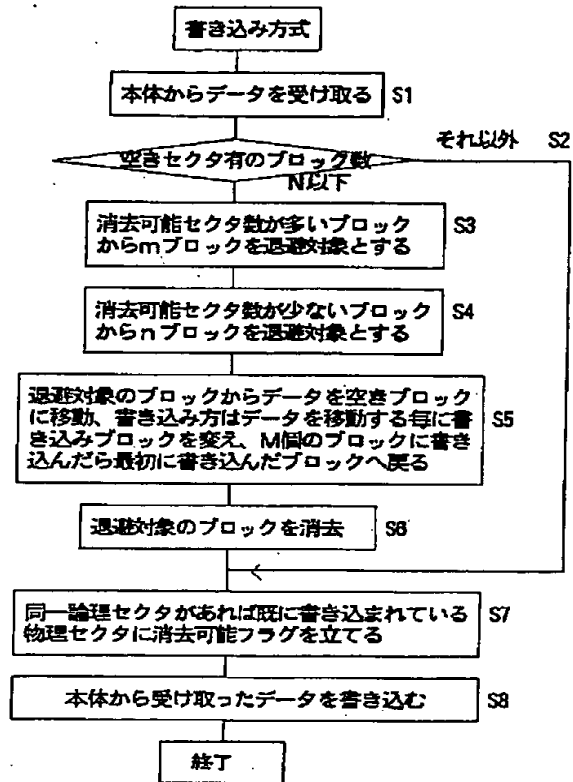
消去回数のバラツキを抑える実施例を示す図（続き）

(ロ) E, F, Gの書き込み

加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
A		消去可	消去可	O	
B		L	N	消去可	
消去可		H	I	K	
消去可		J	G	M	
消去可		E		C	
消去可		F		D	

【図19】

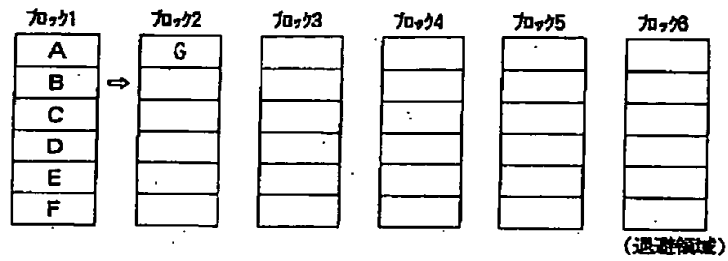
消去回数のバラツキを抑える実施例の処理を示すフローチャート



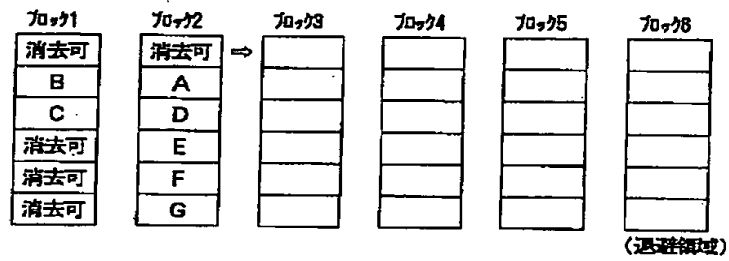
【図20】

消去可能データをなくす処理の実施例を示す図

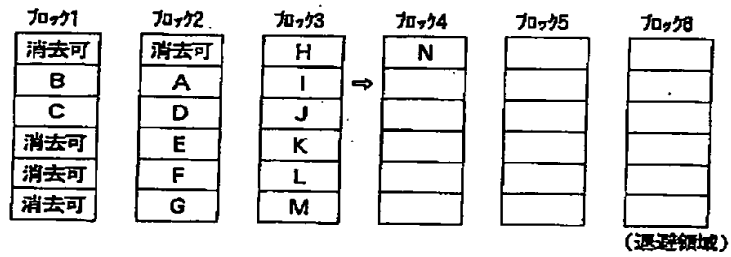
(a) A～Gの書き込み



(b) A、D～Gの書き込み



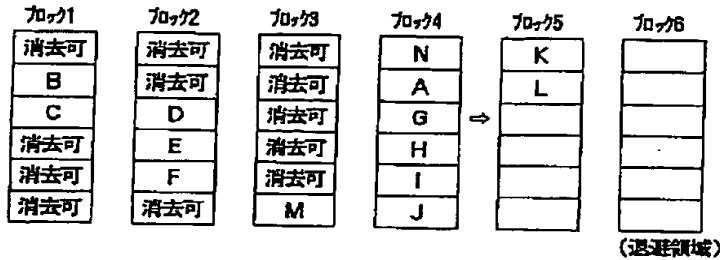
(c) H～Nの書き込み



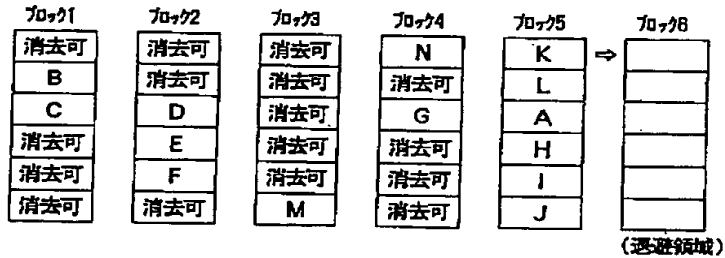
【図21】

消去可能データをなくす処理の実施例を示す図（続き）

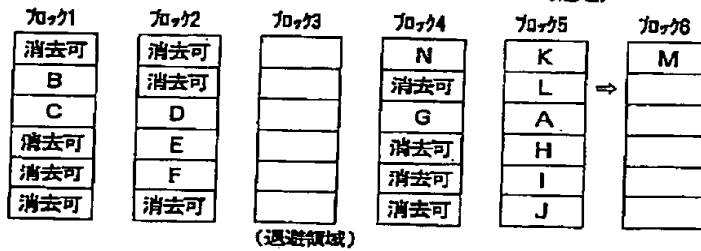
(d) A, G~Jの書き込み



(e) A, H~Jの書き込み



(f-1) A, I~Lの書き込み（書き込み領域がないので退避）



【図22】

消去可能データをなくす処理の実施例を示す図（続き）

(f-2) A. 1~Lの書き込み

加算機1	加算機2	加算機3	加算機4	加算機5	加算機6
消去可	消去可	⇒	N	消去可	M
B	消去可		消去可	消去可	A
C	D		G	消去可	I
消去可	E		消去可	H	J
消去可	F		消去可	消去可	K
消去可	消去可		消去可	消去可	L

(退避領域)

(g-1) 領域処理を行う（ブロック2とブロック3で退避動作）

加算機1	加算機2	加算機3	加算機4	加算機5	加算機6
消去可		D	N	消去可	M
B		E	消去可	消去可	A
C		F	G	消去可	I
消去可	⇒		消去可	H	J
消去可			消去可	消去可	K
消去可			消去可	消去可	L

(退避領域)

(g-2) 領域処理を行う（ブロック1とブロック2で退避動作）

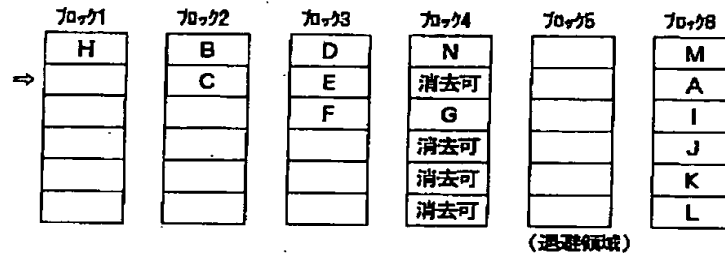
加算機1	加算機2	加算機3	加算機4	加算機5	加算機6
	B	D	N	消去可	M
	C	E	消去可	消去可	A
⇒		F	G	消去可	I
			消去可	H	J
			消去可	消去可	K
			消去可	消去可	L

(退避領域)

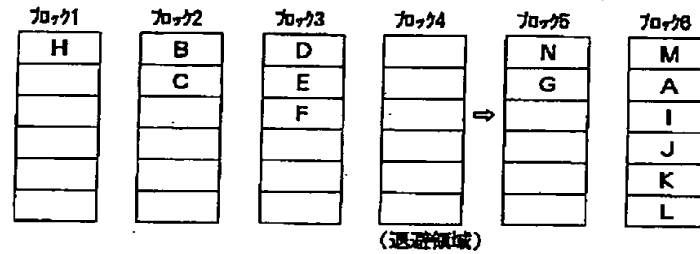
【図23】

消去可能データをなくす処理の実施例を示す図 (続き)

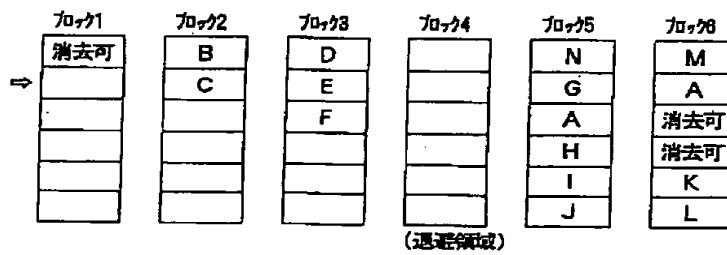
(g-3) 領域処理を行う (ブロック5とブロック1で退避動作)



(g-4) 領域処理を行う (ブロック4とブロック5で退避動作)

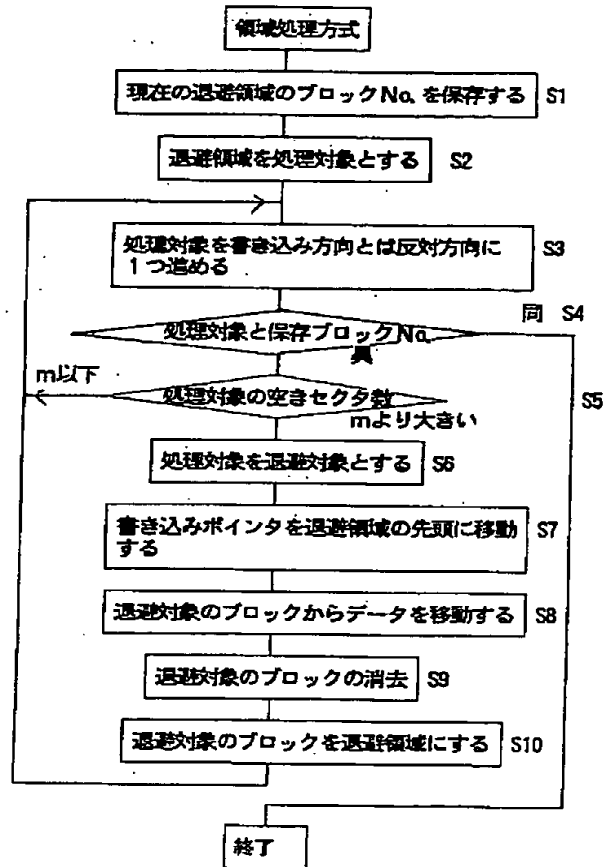


(h) A, H~Jの書き込み



【図24】

消去可能データをなくす処理の実施例を示すフローチャート



【図25】

空き領域の作成処理の実施例を示す図

(a) A～Gの書き込み

70770	70771	70772	70773	70774	70775	70776
	A	G				
	B					
	C					
	D					
	E					
	F					

(予備領域) (退避領域)

(b) A、D～Gの書き込み

70770	70771	70772	70773	70774	70775	70776
	消去可	消去可				
	B	A				
	C	D				
	消去可	E				
	消去可	F				
	消去可	G				

(予備領域) (退避領域)

(c) H～Nの書き込み

70770	70771	70772	70773	70774	70775	70776
	消去可	消去可	H	N		
	B	A	I			
	C	D	J			
	消去可	E	K			
	消去可	F	L			
	消去可	G	M			

(予備領域) (退避領域)

【図27】

空き領域の作成処理の実施例を示す図（続き）

(f-1) A. 1～Lの書き込み（書き込み領域がないので退避動作）

7070	7071	7072	7073	7074	7075	7076
	消去可	消去可		N	K	M
	B	消去可		消去可	L	
	C	D		G	A	
	消去可	E		消去可	H	
	消去可	F		消去可	I	
	消去可	消去可		消去可	J	

(予備領域) (退避領域)

(f-2) A. 1～Lの書き込み

7070	7071	7072	7073	7074	7075	7076
	消去可	消去可		N	消去可	M
	B	消去可		消去可	消去可	A
	C	D		G	消去可	I
	消去可	E		消去可	H	J
	消去可	F		消去可	消去可	K
	消去可	消去可		消去可	消去可	L

(予備領域) (退避領域)

(g) A. M, Nの書き込み（書き込み領域がないので退避動作）

7070	7071	7072	7073	7074	7075	7076
	消去可	消去可	H	N		M
	B	消去可		消去可		A
	C	D		G		I
	消去可	E		消去可		J
	消去可	F		消去可		K
	消去可	消去可		消去可		L

(予備領域) (退避領域)

【図28】

空き領域の作成処理の実施例を示す図（続き）

(h) A, M, Nの書き込み前にブロック5が使用不可となり救済措置

ブロック0	ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
B		消去可	H	N		M
C		消去可		消去可		A
		D		G		I
		E		消去可		J
		F		消去可		K
		消去可		消去可		L

(予備領域) (退避領域)

(使用不可)

(i) 救済措置（予備領域ブロック0からの移動）

ブロック0	ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
		消去可	H	N		M
		消去可	B	消去可		A
		D	C	G		I
		E		消去可		J
		F		消去可		K
		消去可		消去可		L

(予備領域) (退避領域)

(使用不可)

(j) A, M, Nの書き込み

ブロック0	ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
		消去可	H	消去可		消去可
		消去可	B	消去可		消去可
		D	C	G		I
		E	A	消去可		J
		F	M	消去可		K
		消去可	N	消去可		L

(予備領域) (退避領域)

(使用不可)

【図29】

空き領域の作成処理の実施例を示す図（続き）

(k-1) A, G, H, O, Pの書き込み（書き込み領域がないので退避動作）

加 ₁ 0	加 ₁ 1	加 ₁ 2	加 ₁ 3	加 ₁ 4	加 ₁ 5	加 ₁ 6
	G	消去可	H			消去可
		消去可	B			消去可
		D	C			I
		E	A			J
		F	M			K
		消去可	N			L

(予備領域)

(退避領域) (使用不可)

(k-2) A, G, H, O, Pの書き込み

加 ₁ 0	加 ₁ 1	加 ₁ 2	加 ₁ 3	加 ₁ 4	加 ₁ 5	加 ₁ 6
	消去可	消去可	消去可			消去可
	A	消去可	B			消去可
	G	D	C			I
	H	E	消去可			J
	O	F	M			K
	P	消去可	N			L

(予備領域)

(退避領域) (使用不可)

(1) Aの書き込み（書き込み領域がないので退避動作）

加 ₁ 0	加 ₁ 1	加 ₁ 2	加 ₁ 3	加 ₁ 4	加 ₁ 5	加 ₁ 6
	消去可		消去可	D		消去可
	A		B	E		消去可
	G		C	F		I
	H		消去可			J
	O		M			K
	P		N			L

(予備領域)

(退避領域)

(使用不可)

【図30】

空き領域の作成処理の実施例を示す図（続き）

(m) ブロック2が使用不可となり数減措置（空き領域の作成）

加ッ0	加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
B	消去可			D		消去可
C	A			E		消去可
M	G			F		I
N	H					J
	O					K
	P					L

(予備領域)

(使用不可)

(退避領域)

(使用不可)

(n) ブロック2が使用不可となり数減措置（空き領域の作成）

加ッ0	加ッ1	加ッ2	加ッ3	加ッ4	加ッ5	加ッ6
B	消去可		K	D		
C	A		L	E		
M	G			F		
N	H					
I	O					
J	P					

(予備領域)

(使用不可)

(使用不可)

(退避領域)

【図31】

空き領域の作成処理の実施例を示す図（続き）

(o) 救済措置（予備領域ブロック0からの移動）

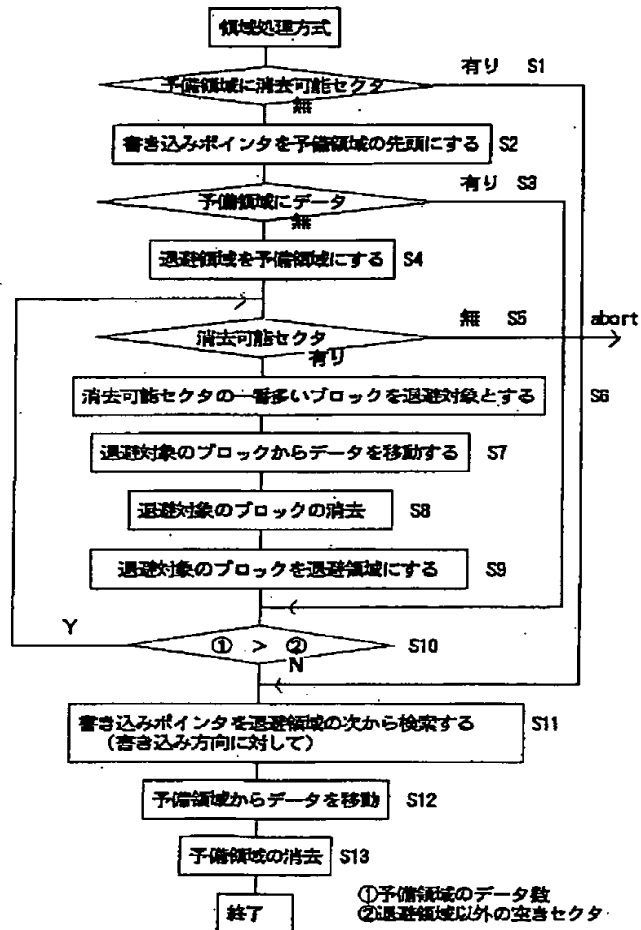
ブロック0	ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
消去可	消去可		K	D		
消去可	A		L	E		
消去可	G		B	F		
消去可	H		C			
I	O		M			
J	P		N			
(予備領域)	(使用不可)			(使用不可)		(逃避領域)

(p) 救済措置（予備領域ブロック0からの移動）

ブロック0	ブロック1	ブロック2	ブロック3	ブロック4	ブロック5	ブロック6
	消去可		K	D		
	A		L	E		
	G		B	F		
	H		C	I		
	O		M	J		
	P		N			
(予備領域)	(使用不可)			(使用不可)		(逃避領域)

【図32】

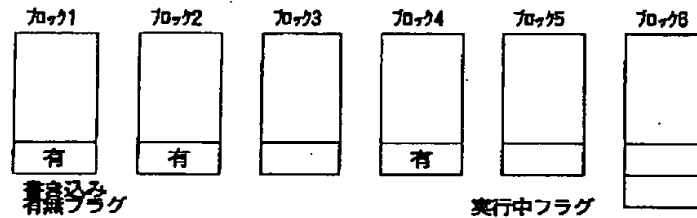
空き領域の作成処理の実施例のフローチャート



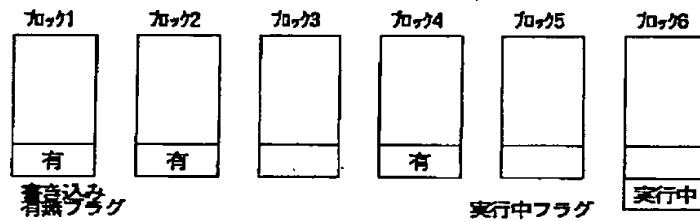
【図33】

消去処理の実施例を示す図

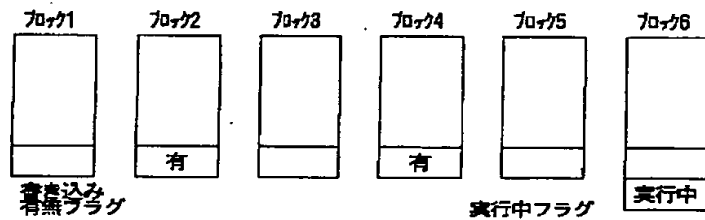
(a) ブロック1, ブロック2, ブロック4の書き込み



(b-1) 初期化 (実行中フラグ書き込み)



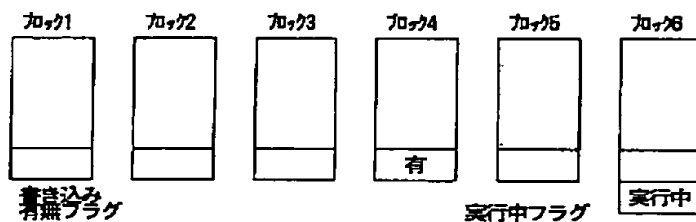
(b-2) ブロック1消去



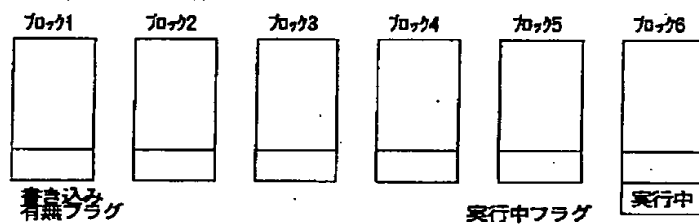
【図34】

消去処理の実施例を示す図（続き）

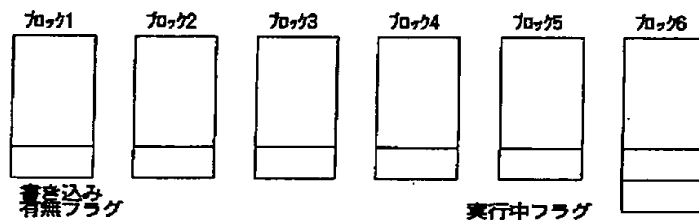
(b-3) ブロック2消去



(b-4) ブロック4消去

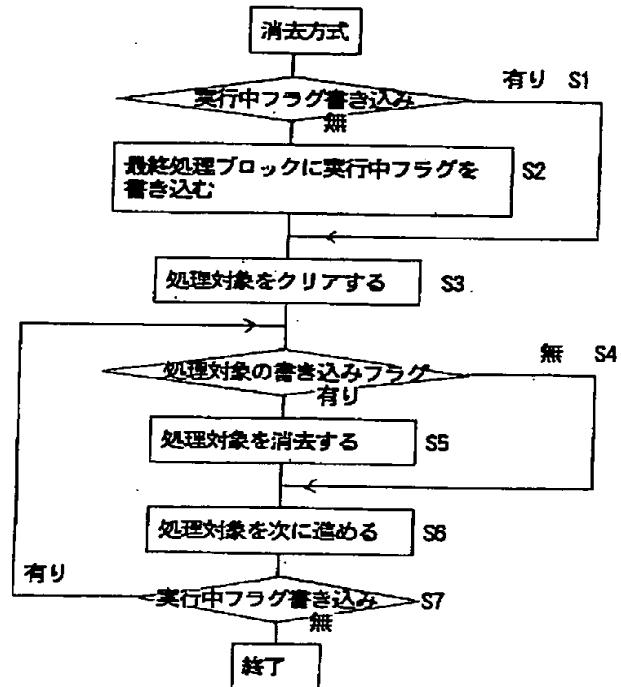


(b-5) ブロック6消去



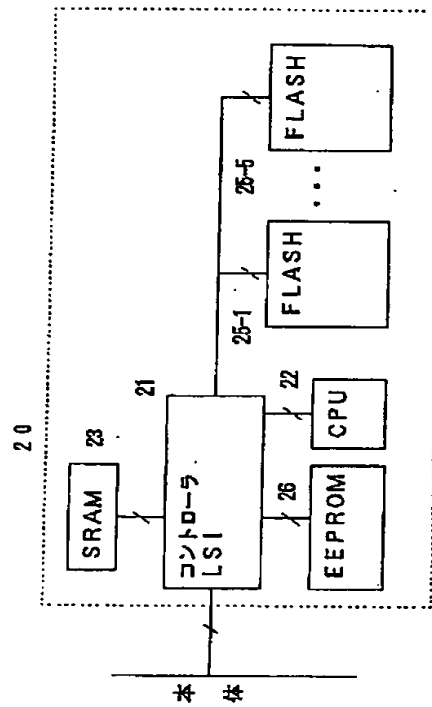
【 図 35 】

消去処理の実施例のフローチャート



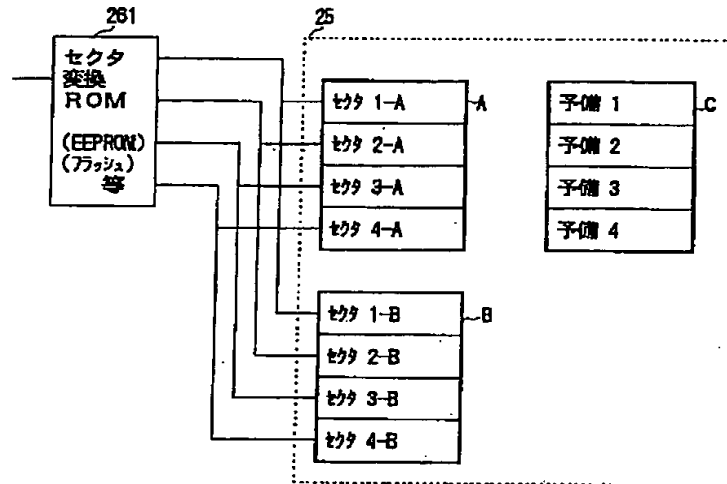
【図36】

書き込み速度の向上を図る実施例のシステム構成を示す図



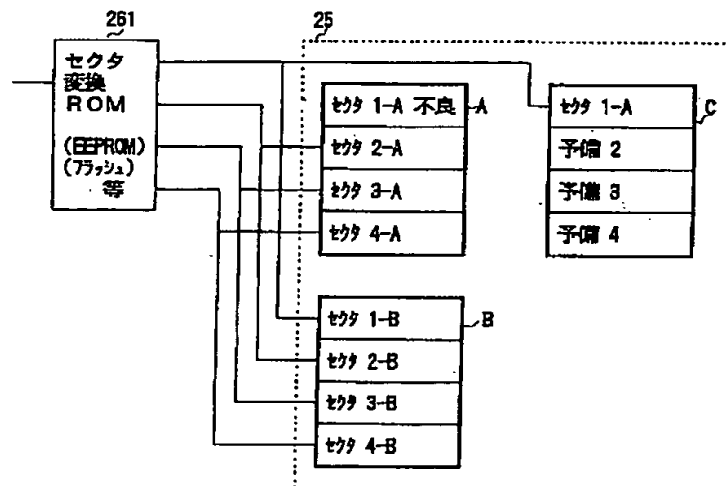
【図37】

書き込み速度の向上を図る第1の実施例を示す図



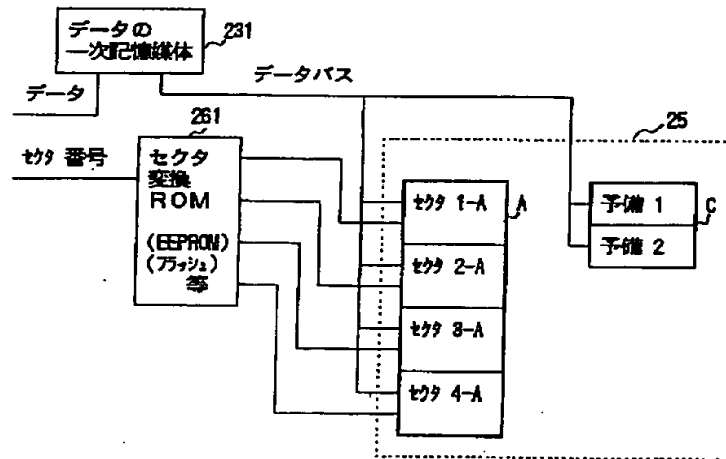
【図38】

書き込み速度の向上を図る第1の実施例を示す図（続き）



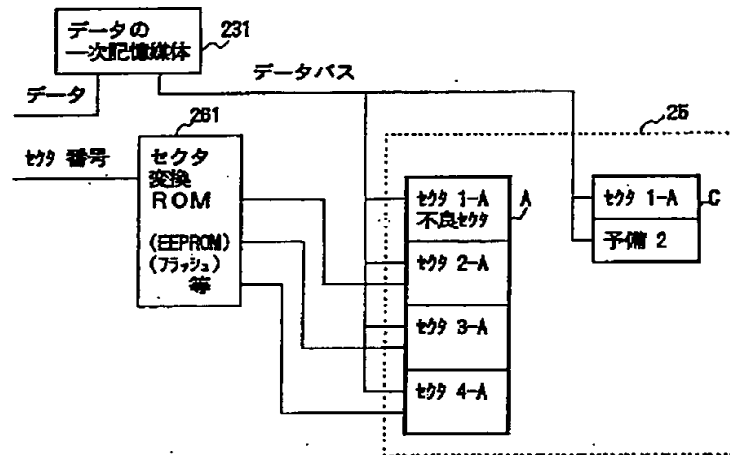
【図39】

書き込み速度の向上を図る第2の実施例を示す図



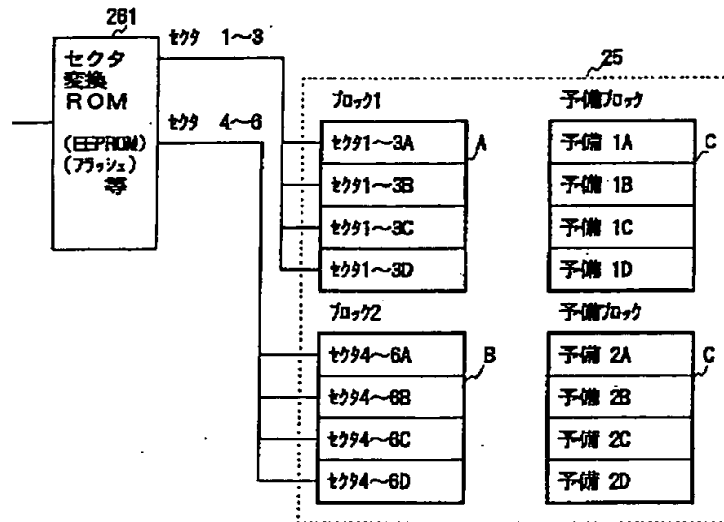
【図40】

書き込み速度の向上を図る第2の実施例を示す図（続き）



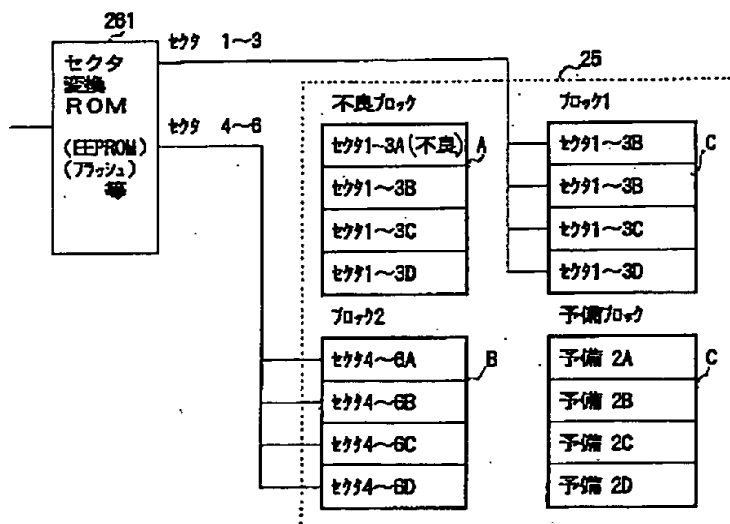
【図41】

書き込み速度の向上を図る第3の実施例を示す図



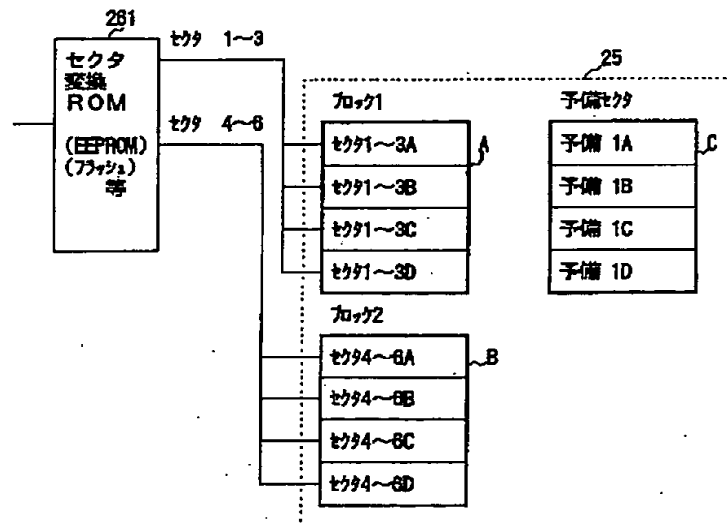
【図42】

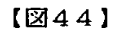
書き込み速度の向上を図る第3の実施例を示す図（続き）



【図43】

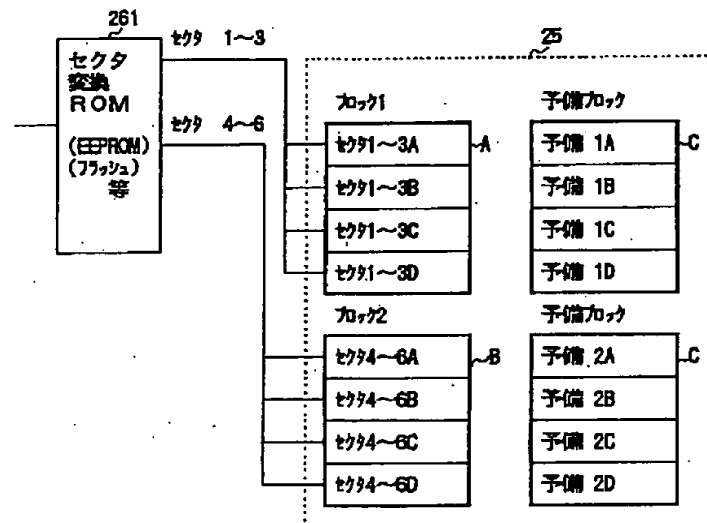
書き込み速度の向上を図る第4の実施例を示す図





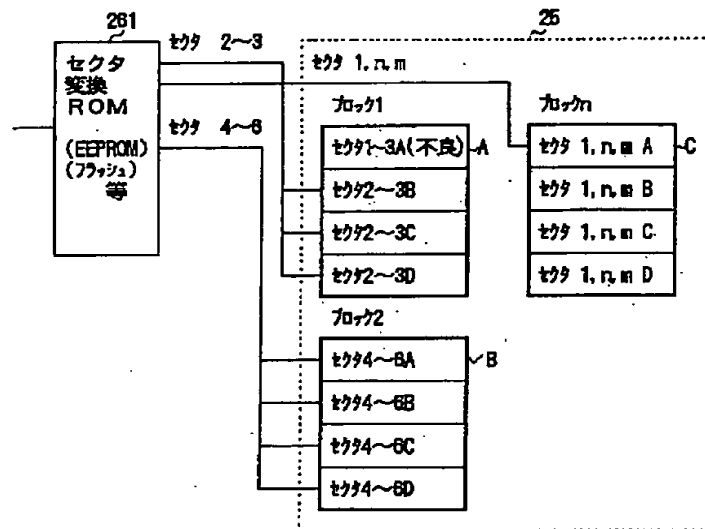
【図45】

書き込み速度の向上を図る第5の実施例を示す図



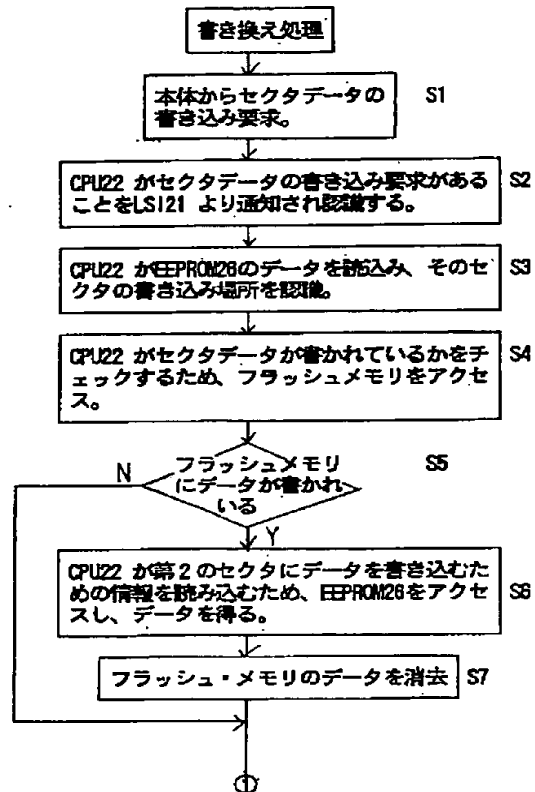
【図46】

書き込み速度の向上を図る第5の実施例を示す図（続き）



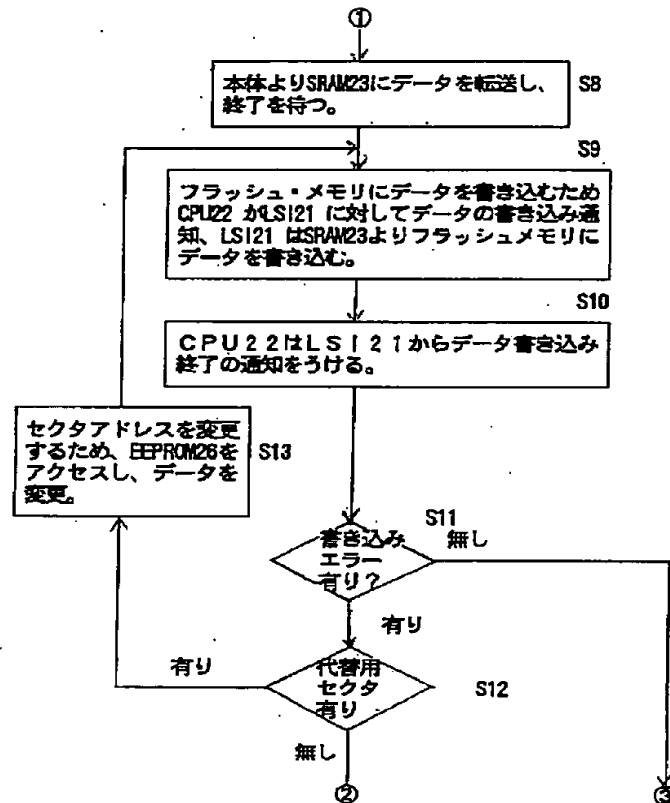
【図47】

書き込み速度の向上を図る実施例のフローチャート



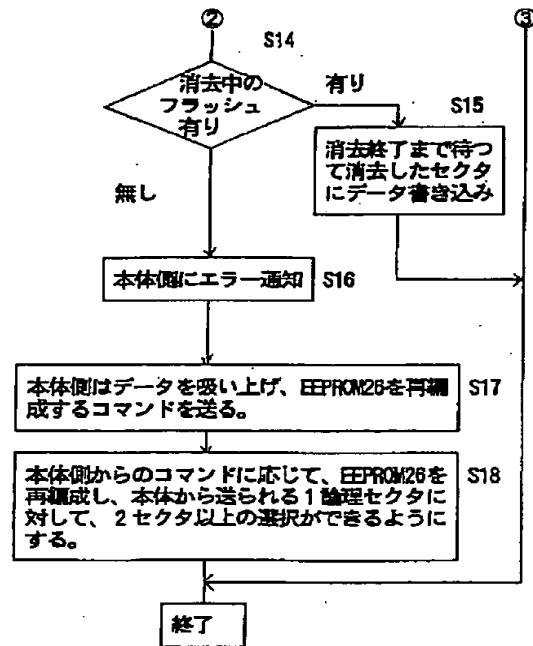
【図48】

書き込み速度の向上を図る実施例のフローチャート（続き）



【図49】

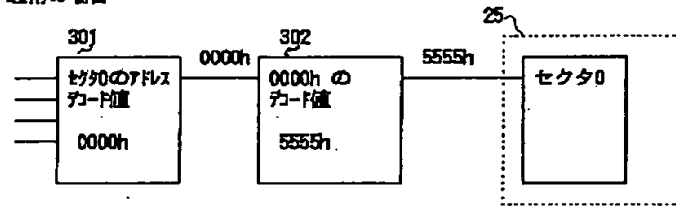
書き込み速度の向上を図る実施例のフローチャート（続き）



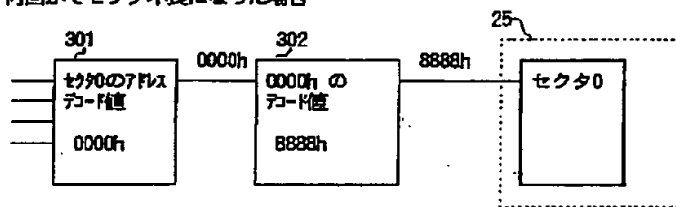
【図50】

アドレス変換用デコード部を2段とした実施例を示す図

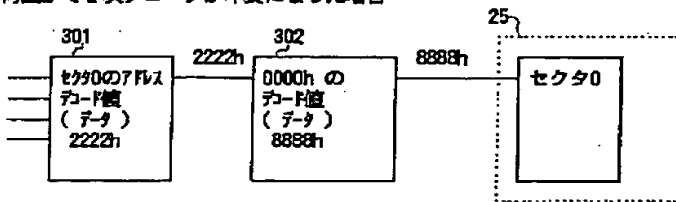
(a) 通常の場合



(b) 何回かでセクタ不良になった場合

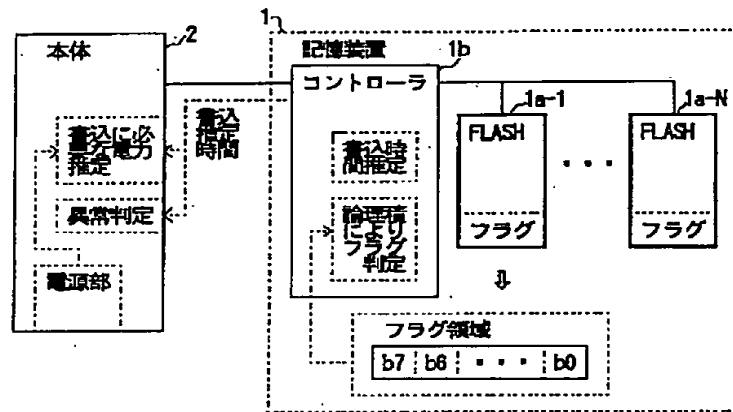


(c) 何回かで2次デコーダが不良になった場合



【図51】

書き込み時間推定処理を行うシステムの全体の概略構成を示す図



【図52】

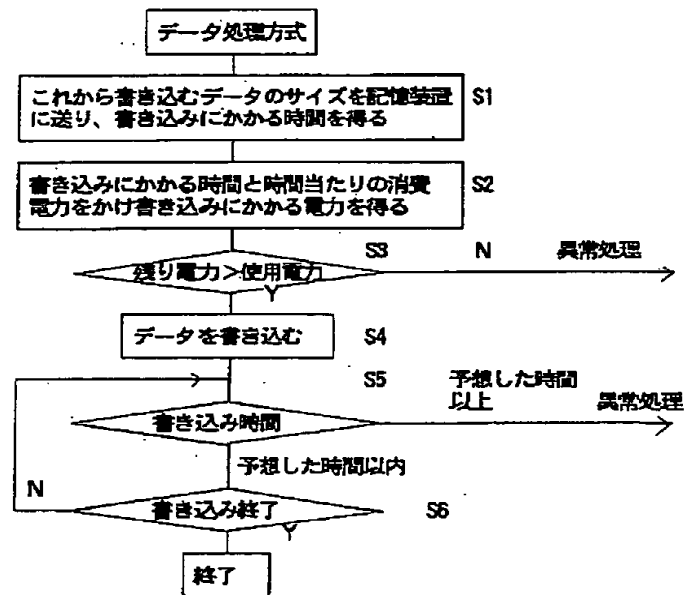
書き込み時間推定処理の実施例における記憶領域の状態を示す図

7071	7072	7073	7074	7075	7076
A	G	F	消去可	消去可	
消去可	消去可	H	消去可	K	
消去可	消去可	消去可	N	L	
消去可	C	B	消去可	M	
消去可	D	消去可	I		
消去可	E	消去可	J		

(書き込み領域)

【図53】

書き込み時間推定処理の実施例のフローチャート



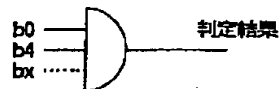
【図54】

フラグ判定処理の第1の実施例を示す図

(a) フラグレジスタ



(b) フラグ判定方法



(c) 真値値表

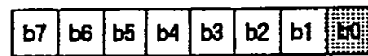
行番	b4	b0	結果
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

⇐ (b0ビットが1 固定不良となった場合)
⇐ (b4ビットが1 固定不良となった場合)

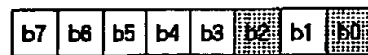
【図55】

フラグ判定処理の第2の実施例を示す図

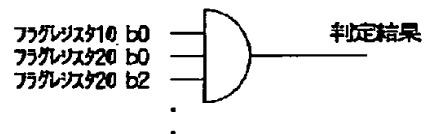
(a) フラグレジスタ1



(b) フラグレジスタ2

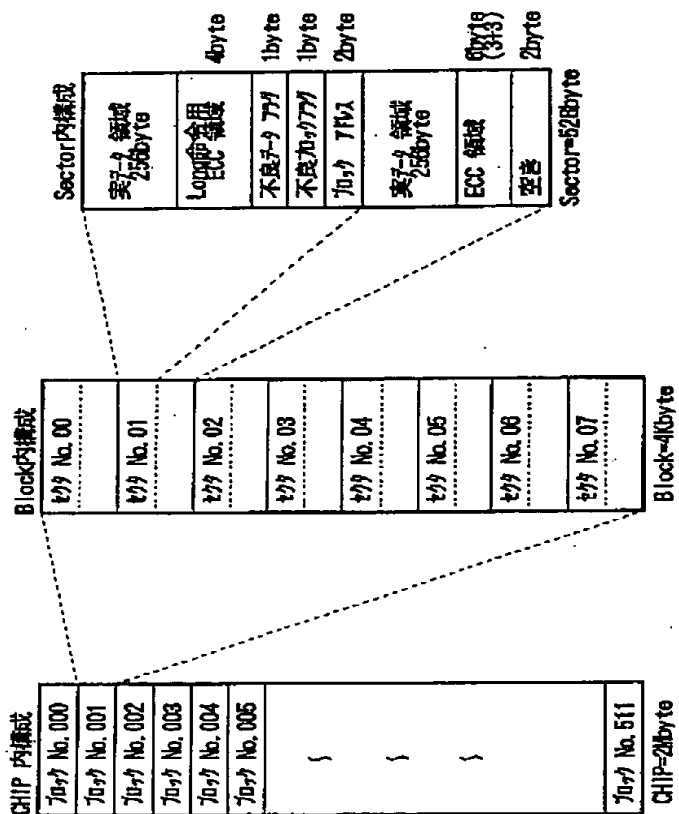


(c) 判定方法



【図56】

管理テーブルを節約する実施例における記憶領域の構成を示す図



(a) 全消去後

[illegible]

CHIP No. 04
032 ~039sec
152 ~159sec
Work03
Work04
Work01
Work02
.
.

	CHIP No. 03
	Q24 ~ Q31/sec
	144 ~ 151/sec
	184 ~ 191/sec
	Work04
	Work01
	Work02
	Work03
	.
	.

CHIP No. 02
-016 ~-023sec
136 ~143sec
176 ~183sec
Word04
Word01
Word02
Word03
.
.

CHIP No. 01
008 ~ 016sec
128 ~ 135sec
168 ~ 175sec
Work04
Work01
Work02
Work03
.
.

CHIP No. 00	000 ~007sec
	120 ~127sec
	180 ~187sec
	World04
	World01
	World02
	World03
	:
	:

CHIP No. 04
032 ~089sec
162 ~169sec
Work03
Work04
Work01
Work02
.
.

CHIP No. 03
024 ~03 sec
144 ~15 sec
184 ~19 sec
Work04
Work01
Work02
Work03
.
.

CHIP No. 02
016 ~023sec
180 ~143sec
179 ~183sec
Work04
Work01
Work02
Work03
.
.

CHIP No. 01
008 ~015sec
128 ~135sec
168 ~175sec
Work04
Work01
Work02
Work03
.
.

CHIP No. 00
Work04
120 ~127sec
160 ~167sec
000 ~007sec
Work01
Work02
Work03
.
.

(d) 2セクタ選挙書き込み (論理アドレス: セクタNo. 002~003)

【図61】

管理テーブルを節約する実施例を示す図 (続き)

(e) 2セクタ連続書き込み後電源OFF→ON	CHIP No. 00				CHIP No. 01				CHIP No. 02				CHIP No. 03				CHIP No. 04			

フロントページの続き

(51) Int. Cl.⁷
G11C 29/00識別記号
601
631
671FI
G11C 29/00
17/00

テームト (参考)

671B
601C
601B
633A
639B
639C

(72)発明者 林 朋弘
神奈川県横浜市中区本町4丁目36番地 株
式会社富士通コンピュータテクノロジ内
(72)発明者 柴崎 省吾
神奈川県横浜市中区本町4丁目36番地 株
式会社富士通コンピュータテクノロジ内

(72)発明者 伊藤 裕之
神奈川県横浜市中区本町4丁目36番地 株
式会社富士通コンピュータテクノロジ内
(72)発明者 竹原 勝
神奈川県横浜市中区本町4丁目36番地 株
式会社富士通コンピュータテクノロジ内